2020

Topological Analysis of Averaged Sentence Embeddings

Wesley J. Holmes
Wright State University

Topological Analysis of Averaged Sentence Embeddings

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

by

WESLEY J. HOLMES
B.S.C.S., Wright State University, 2018

2020
WRIGHT STATE UNIVERSITY

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

December 1, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Wesley J. Holmes ENTITLED Topological Analysis of Averaged Sentence Embeddings BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Michael Raymer, Ph.D.
Thesis Director

Mateen M. Rizki, Ph.D.
Chair, Department of Computer Science and Engineering

Committee on
Final Examination

Michael Raymer, Ph.D.

Mateen Rizki, Ph.D.

Krishnaprasad Thirunarayan, Ph.D.

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

ABSTRACT

Holmes, Wesley J. M.S., Department of Computer Science, WRIGHT STATE UNIVERSITY, 2020. *Topological Analysis of Averaged Sentence Embeddings.*

Sentence embeddings are frequently generated by using complex, pretrained models that were trained on a very general corpus of data. This thesis explores a potential alternative method for generating high-quality sentence embeddings for highly specialized corpora in an efficient manner.

A framework for visualizing and analyzing sentence embeddings is developed to help assess the quality of sentence embeddings for a highly specialized corpus of documents related to the 2019 coronavirus epidemic.

A Topological Data Analysis (TDA) technique is explored as an alternative method for grouping embeddings for document clustering and topic modeling tasks and is compared to a simple clustering method for effectiveness.

The sentence embeddings generated are found to be effective for use in similarity based tasks and group in useful ways when used with the TDA based techniques explored as alternatives to traditional clustering-based approaches.

Contents

List of Figures

List of Tables

Acknowledgment

I would like to take this opportunity to extend my thanks to my Advisor, Dr. Raymer, for his valuable guidance and constructive criticism, which helped to bring my writing to a higher level.

I would also like to thank my committee members, Dr. Prasad and Dr. Rizki, for their additional guidance and feedback.

Additionally, I would like to thank my parents for their constant support and encouragement. Without them I would not have completed this thesis.

Finally, I would like to thank my friends Pete and Howey for their support and their ability to distract me from the stress of writing.

Dedicated to

My Mother

Introduction

Natural language processing, or NLP, is the field of using computational algorithms and techniques to decipher and process text written by humans. Because of the complexity of working with unstructured text, NLP tasks are typically decomposed into simpler subtasks that are then composed into pipelines for end-user applications. Subtasks can be straightforward, such as stemming, stopword elimination, and word tokenization, or more challenging such as part-of-speech recognition, phrase extraction, named entity recognition, and generation of character or word embeddings. End-user applications leverage these subtasks together with language models to perform tasks such as document clustering and retrieval, semantic search, named-entity recognition, keyphrase extraction, and document and/or corpus summarization. Document clustering and retrieval are areas of particular interest due to their uses for querying large corpora of data, as done by search engines. Topic modeling is another useful area as it allows for analysis of the content of documents and discovery of trends over time in a corpus. This thesis applies topological data analysis to a current method of generating sentence embeddings. I propose that a topological-analysis-based hierarchy of sentences based on averaged word embeddings will reflect the semantic relationship among sentences in a hierarchical fashion and facilitate identification of sentences with similar meaning at varying levels of abstraction. This hierarchy allows for effective clustering of documents, and for querying a large corpora of specialized data for specific topics and trends. While most techniques for document clustering and topic analysis are focused on large corpora or a variety of wide-ranging topics, the system in this paper is

designed for smaller corpora of scientific research publications. In the case of this paper, the system was built using the CORD-19 research dataset [39], although these techniques can be expected to generalize to other similar corpora.

Related Work

Words are an inherently difficult set of data for computers to work with, words with similar spellings can have very different meanings, such as boat and coat. These words differ only in their first letter, but a boat is a buoyant vehicle, and a coat is an article of clothing. Additionally, the same word can have vastly different meanings depending on the context. For example, the word club can have one of several different meanings depending on its context. In one sentence, club could be referring to a type of sandwich. In another, the sentence could be talking about an extracurricular meeting group. In yet another context, club could be referring to a piece of golf equipment

In addition to words that appear to be similar having entirely different meanings, there are also words that look nothing alike that have very similar meanings. All of these potential difficulties for understanding the meaning of a word make it necessary to have a more meaningful representation for more complex tasks.One approach is to embed each word into a metric space of dimensionality, k, such that each work is represented by a numerical vector of length k. These representations, or word embeddings, can be generated in a variety of ways, discussed later. These word embeddings are designed so that their similarity in the vector space captures as much of the semantic meaning and similarity of the words themselves. For example subtracting the representation for the word man from the representation for the word king and adding the representation for woman should result in a vector very close to, if not the same as the representation for the word queen. These word embeddings have been found to be an effective representation for words and are widely

used across NLP fields.

A language model is a probabilistic model of word occurrence given other words in a sequence. These can be divided into two general types, forward models, and bidirectional models. Forward models are often used for auto-completion tasks, and attempt to predict words based on the preceding words in a sequence. Bidirectional models both preceding words in a sequence and words following the word to predict. These models can be used for a variety of tasks, but are often used to generate representations, where accurate predictions indicate more accurate representations for the sequence or word being embedded.

An n-gram model is a language model that divides a body of text into smaller components. These components, n-grams, consist of a sequence of n individual units from the text. These can be words when using the technique for larger bodies of text such as sentences or documents. They can also be individual characters when the models are applied to individual words in a document. A skip-gram model is a model used for generating word embeddings that attempts to learn an embedding for a word based on the words surrounding it. These models are an effective method for generating word embeddings and are discussed more in depth later.

One-hot-vector word representations are a very simple method for generating word embeddings. In these embeddings, the vector is as many dimensions as there are in a vocabulary. Each word in the vocabulary receives a vector where one of the entries has a value of 1, and the rest have a value of 0. This approach can be extended to documents, where a document has a value at each position corresponding to a word that occurs in a document. Bag of words representations treat documents as a collection of words, with no respect to ordering or proximity, similar to a bag. The contents of the document are maintained, but ordering and proximity relationships are not maintained. These approaches are a commonly used method for word embeddings, that are trained on the sentence level. Due to the lack of ordering, some semantic meaning is lost when using this type of model to generate word embeddings.

There are a wide range of techniques used to generate embeddings for words, and to a lesser extent, sentences. One of the most common methods for generating word embeddings is Word2Vec [27], which uses a continuous bag of words model that trains a model to predict a word based on the surrounding context words. A continuous bag of words is similar to a bag of words model, except that the words are continuously updated, as a result only a small subset of the document is used at a given time. Similar approaches include fastText [6], which also uses a continuous bag of words approach, but also includes subword information when training a word. In addition to these relatively simple models there are much more complex models trained on very large corpora such as GloVe [32], BERT [10] and its derivatives like bioBERT [21]. These models are available pre-trained to generate word embeddings. Pre-taining is generally performed on a general corpus, such as documents collected from wikipedia articles at a particular point in time.

For sentence embeddings there are two general techniques, either combinations of word embeddings for sentences or more complex models that generate sentence embeddings directly, commonly based on models such as BERT [34]. For simple models, Weiting, et al. note that simple averaging of word embeddings for all words in a sentence performs surprisingly well for many tasks [40]. Further, Arora, et al. discovered that performing weighted averaging provides an increase in performance as well [3]. The disadvantage that these techniques have is that the ordering of words in a sentence has no impact on the embedding generated. More complex neural models provide a solution to this problem, but again at the disadvantage of being very large and complex, and with minimal ability to retrain on a specific corpus.

In the realm of document clustering and classification, many techniques use term frequency based approaches where a document has a vector representation based on the terms in the document, as well as how often those terms appear in other documents in a corpus.

In the realm of topic modeling the most commonly used techniques are based on Latent Dirichlet Allocation [5], with modifications based on the specific domain and goal.

LDA assumes that topics can be viewed as distributions, and documents can consist of a distribution of topics.

In the realm of keyphrase extraction there are a variety of techniques being used, with some based on term frequency based approaches, as well as a few methods using topological methods.

2.1 Word Embeddings

Word embeddings are a commonly used technique to convert text data into a format that can be used in more complex NLP systems. Many of the most popular word embedding models are based on neural networks, and work to optimize a vector representation for words in a corpus. In general, neural networks consist of a series of nodes that perform a weighted sum of inputs and apply a nonlinear function to the sum to obtain an output. The weights are optimized over multiple inputs to minimize the difference between the output of the neural network and the expected value. A variety of different techniques exist for training word embedding models, with two major overarching types; models that focus on local context information such as Word2Vec [27] and fastText [6], and models that focus on more global information for words such as GloVe [32]. These techniques are widely used, and pretrained models are available for general purpose use for all three of them. These models are also simple enough that most users can train them for a specific corpus if it is required. In addition to these simple models, there are also much more complex models such as BERT and its derivatives, which use very complex architectures trained on much larger data sets. These models are almost always used as pretrained models due to the large amount of data needed to train them from scratch.

2.1.1 GloVe Embeddings

Global Vectors for Word Representation (GloVe) is a newer method for generating word embedding vectors, developed by Pennington et al [32]. While the most commonly used methods for generating word embedding vectors are based on training supervised neural networks to minimize loss in a local context, GloVe is an unsupervised method that focuses on the global statistics of words in a corpus. The basis for GloVe is a word to word co-occurrence matrix. This matrix is calculated by counting how many times a pair of words occur in the same context, such as a sentence or smaller grouping of words. Each word in the co-occurrence matrix is weighted by all other words in the corpus to obtain a vector representation that captures the probabilities of other words occurring in the sentence. This vector is then reduced in size to 100 dimensions and is used as the final embedding vector for a word. In their work, Pennington et al found that GloVe vectors perform better than other statistical word embedding models for word analogy tasks, with some improvement over Word2Vec representations with the same dimensionality as well [32].

2.1.2 Word2Vec Embeddings

Word2Vec is a commonly used method for generating word embedding vectors. It is available in two different varieties, skip-gram [28] and continuous bag of words (CBOW) models [27]. Both types of models generate a word embedding based on the surrounding words or context. A skip-gram model uses a word, and attempts to accurately predict the words in a window surrounding it. As shown in Figure 2.1, the input to a skip-gram model is the current vector representation for a word, and the outputs are the predicted embeddings for the surrounding words [28]. These outputs are compared to the current embeddings and the model attempts to predict them as accurately as possible.

A CBOW model uses an almost opposite approach, and attempts to predict a word based on the context words surrounding it. The inputs to a CBOW model are the current

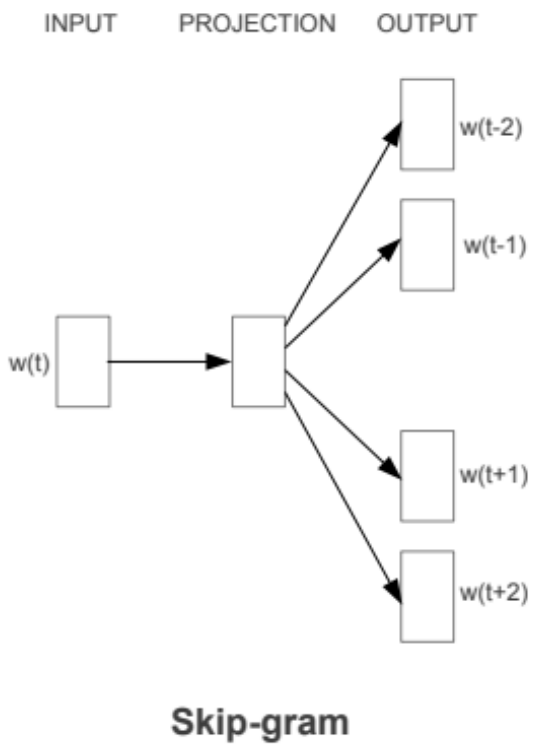

Figure 2.1: A visual representation of a Skip-gram model. From [28].

vectors for the words surrounding the word being learned, with an even number before and after, as shown in Figure 2.2. The model attempts to correctly predict the vector for the output word by performing a weighted sum of the input vectors.

The embeddings generated by both models have been found to be effective for many different tasks such as text classification, especially when combined with other techniques, [22] and finding analogous words [32]. It has been found that models using a skip-gram architecture perform better on tasks requiring semantic information, though other models such as fastText outperform even those models. An interesting property of Word2Vec representations of words is that vector arithmetic can be applied to them to get accurate approximations of other words in a corpus [27]. For example, subtracting the embedding for man from the embedding for king and adding the embedding for woman results in a vector very close to the vector for the word queen. This property suggests that the embedding

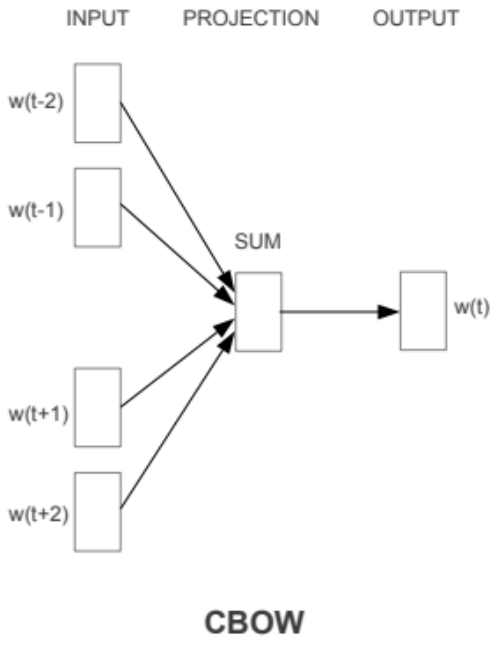

Figure 2.2: A visual representation of a CBOW model. From [28].

vectors exist in a meaningful space that may be analyzed using topological data analysis techniques. In addition to being easy to understand and potentially visualize, Weiting et al, found that averaging word embeddings from Word2Vec can be an easy and effective method for generating sentence embeddings [40]. This work was further improved on by Arora et al. by performing weighted averaging of the words in a sentence, which was found to improve their effectiveness for some tasks such as evaluating semantic similarity [3].

2.1.3 fastText Embeddings

FastText is a method for generating word embedding vectors developed by Facebook [6]. It is similar in practice to Word2Vec in that it is an unsupervised approach that uses a relatively simple method to train a model. FastText supports using a continuous bag of words model or a skip-gram model to obtain word embeddings. A continuous bag of words model uses a sliding window over sentences and attempts to accurately predict a missing word in the window based on the remaining words in the window. A skip-gram model also uses

a sliding window over the sentences, but instead uses a word in the window to predict the other words within the window. Skip-gram models require a larger training corpus to generate effective embeddings, but generally capture the semantic meaning of words better than continuous bag of words models. In addition to using the surrounding words to generate an embedding for a word, fastText also uses subword information, meaning small character sequences that are shorter than the length of the word. This inclusion of subword information is the main difference between fastText and Word2Vec, and helps to improve the model's ability to capture semantic meaning in the embeddings. The subword information also allows fastText models to approximate an embedding for words not in an initial training vocabulary as long as at least one subword component was already included in a training word. FastText models have been found to be effective when used for both semantic analysis tasks, as well as text classification problems [17]. These models have been found to perform as well as more complex deep learning models, and in some cases outperform them for certain tasks [17]. The ability to handle words not in a training vocabulary, and to perform as well as more complex deep learning models while requiring smaller corpora to obtain accurate embeddings make this method ideal for obtaining accurate word embeddings for tasks that require user input. In addition these models work much better on smaller datasets.

2.1.4 BERT Embeddings

Bidirectional Encoder Representations from Transformers models are deep learning models for generating word embeddings based on bidirectional transformers [10]. These models are primarily focused on learning local context to generate word embeddings. As opposed to Word2Vec and fastText models, these models read an entire input sentence at a time, instead of reading the sentence from left to right and viewing individual windows of the sentence consecutively. The model attempts to accurately predict the next sentence in the document based on the entire current input sentence. This allows the model to capture con-

text information that may be outside of the smaller context window sizes used by fastText and Word2Vec. In addition, the usage of the entire sentence at once allows for capturing of context that may be relevant for a word at the beginning of the sentence from words near the end of the sentence. This basis of looking right to left in addition to left to right as in context window methods is the largest difference between the two model types.

Due to the complex nature of these models, most are available as pretrained models due to the fact that they require much larger corpora to train than models such as Word2Vec and fastText. These pretrained models are widely used with few instances of models being trained de novo on individual corpora. The complex nature of these models also results in higher hardware requirements for training than Word2Vec and fastText models. While the original BERT model was trained on a corpus based on wikipedia data, further work has been done to produce models for more specific tasks such as SciBERT and bioBERT. SciBERT uses the same underlying architecture as the original BERT model, but is trained on a corpus containing only scientific documents, and as such performs better on tasks related to scientific domains [4]. In addition to SciBERt, an even more specialized version was developed, trained only on biomedical documents. This version of BERT is known as bioBERT, and outperforms both BERT and SciBERT in tasks using text from biomedical and biological fields [21].

Due to the fact that these models look at entire sentences at a time some work has been done to use pretrained models to directly produce sentence embeddings by making slight modifications to the architecture of later layers and performing small amounts of additional training. This additional training is often task and corpus specific and is commonly referred to as tuning the model. These models are discussed more in depth in

2.2 Sentence Embeddings

There are two primary strategies that are used to generate sentence embeddings, averaging approaches based on combining individual word embeddings, and complex deep learning architectures that attempt to generate sentence embeddings directly, with no intermediate word embedding step. Naive averaging approaches have been shown to be effective at textual similarity and classification tasks [3], and are usually much cheaper to compute than the embeddings produced by deep learning approaches. Deep learning approaches, while more costly, generally have better performance in most tasks, though the level of improvement varies, with more improvement noticeable in tasks where word order is very important.

2.2.1 Averaging Methods

As mentioned earlier, Weiting et al, found that averaging Word2Vec embeddings for a sentence provided embeddings that were effective across a variety of tasks. These results can be used as a baseline as the embeddings were based on a pretrained Word2Vec model with no topic specificity, which may drastically change results [3]. These findings were further improved by Arora et al, who found that performing weighted averaging of the word embeddings helped to improve performance in semantic similarity and text classification tasks [40]. Based on this information it is likely that more major changes, such as training a word embedding model on a specialized corpus for different tasks would likely perform even better than using a generic pretrained model. These approaches to generating sentence embeddings are not without their issues as they lack the ability to capture semantic meaning as well as other more complex deep learning approaches. In addition, the ordering of words in a sentence does not affect the resulting embedding, which can present issues for tasks where word ordering is more important, as shown in [1]. In order to alleviate some of the issues with the lack of semantic meaning being retained, it may be possible to perform

averaging of word embedding vectors that retain more semantic meaning, such as fastText or GloVe embeddings as opposed to those generated by Word2Vec. While these approaches show promising results, most work in the area of generating sentence embeddings has been focused on using more complex models such as deep learning models.

2.2.2 Sent2Vec

A similar method for generating sentence embeddings is Sent2Vec. This approach to generating sentence embeddings is an extension of the original Word2Vec model developed by Mikolov, but instead of generating word embeddings, and then averaging them after a word embedding model has been trained, the embeddings are averaged during training. In addition to generating word embeddings, n-gram embeddings for each sentence are included in the averaging step for generating the sentence embeddings. This model provides improved performance in textual similarity tasks over simple averaging techniques [30]. Though this technique provides improved performance over Word2Vec models, it requires a much longer time to train the model, even when trained on a GPU [30]. This technique also allows for training models in a supervised manner, which results in further improved performance, and acts as an intermediate model between the more simple averaging methods and the more complex deep learning architectures.

Sent2Vec has been modified in a similar way to BERT in that there has been work to make pretrained models for more specialized tasks available. One of these efforts is BioSentVec, which uses the Sent2Vec architecture, but trains on a specialized corpus of biomedical data [9]. This model improves on the performance of Sent2Vec in tasks related to biomedical documents [9].

2.2.3 Deep Learning Methods

While there are a variety of deep learning approaches to generating sentence embeddings available, a large portion are based on transformers, more specifically the Bidirectional Encoder Representations from Transformers (BERT) model developed by Devlin et al [10]. This model, and more topic specialized derivatives such as bioBERT and sciBERT are frequently used to generate sentence embeddings. These models are much more complex than the models used to generate word embeddings for averaging approaches. These models use bidirectional information, meaning sentences are read forward and backward, resulting in more context information being captured. In addition, embeddings for individual words in these models can be different based on the surrounding context words in a sentence. Due to the much higher complexity of these models, they are often only used as pretrained models that have been trained by other groups on very large corpora. This prevents them from being used as effectively in very specialized domains, as these corpora are rarely large enough to train these models. In addition to the need for large corpora, even generating embeddings on a pretrained model is extremely resource and time intensive.

While BERT models are primarily used to generate word embeddings, and these word embeddings can be combined using the same methods that other word embeddings can be combined with, some work has been done to modify these models to produce sentence embeddings directly. The leading effort for this is sentence-BERT, which modifies the architecture of the models to directly output sentence embeddings [34]. These modifications are minor, and allow for the use of pre-trained BERT models to generate sentence embeddings, meaning bioBERT and sciBERT models can be used to generate sentence embeddings without any new training. These modifications work and do not require re-training of the model as the training method for the initial model is to predict the next sentence in a document, which is sufficient for generating sentence embeddings. As these embeddings come from internal layers of more complex models, the resulting embeddings may be less understandable to an end user, and may also not have the beneficial vector properties of

those generated by Word2Vec and fastText models.

2.3 Document Embedding and Clustering

In addition to generating embeddings for words and sentences, there has been some work to develop effective compressed representations for entire documents in a corpus, often for the purpose of document clustering, which is the process of grouping documents that are similar. The primary methodology for generating document embeddings are techniques based on term frequency-inverse document frequency (TF-IDF) [33].

2.3.1 TF-IDF

TF-IDF is the most common method for generating compressed document representations. TF-IDF begins by selecting a vocabulary of words to be included, which is usually much smaller than the entire vocabulary in the corpus. After selecting a vocabulary, every word in the vocabulary has its term frequency and document frequency calculated. Term frequency is the count of occurrences of the word in a given document. Document frequency is the number of documents in the corpus a word occurs in. TF-IDF is calculated for each word in each document as term frequency times the log of 1 over the document frequency. After calculating the TF-IDF for every word, a document can be represented as a vector of the TF-IDF values for all words in the vocabulary. This method is a global method with respect to the corpus of data, meaning that the embedding for a document depends on every other document in a corpus, and adding a new document would require recalculating the TF-IDF for every other document in the corpus already. In addition to the global nature of the resulting embeddings, the vocabulary used for generating the embeddings often cannot include all words in the corpus, as vectors are sparse to begin with, and inclusion of all words would result in unreasonably sized vectors.

TF-IDF representations for documents have been used as the representation for documents in a variety of document clustering techniques and are often used for most document retrieval algorithms [31].

While TF-IDF on its own is an effective method for generating embeddings for documents, there are related techniques that add supplemental information to the resulting vectors for documents in an effort to improve performance in various tasks.

2.3.2 Embedding Averages

Similarly to generating sentence embeddings by averaging word embeddings, some work has been done to generate document embeddings by averaging word embeddings, as well as averaging sentence embeddings. This approach is very rare, and still in preliminary stages, as TF-IDF vectors frequently perform sufficiently well for most tasks, and are well established as an effective method. The most commonly used technique based on averaging is Doc2Vec [20]. Doc2Vec functions similarly to Word2Vec and Sent2Vec, except that it attempts to generate embeddings for entire paragraphs, which can then be combined to generate an embedding for an entire document.

2.3.3 Document Clustering

The objective of document clustering is to group documents that appear similar, either usually by their content. Document clustering is often used for document retrieval, and in search engines to return results that may be relevant based on a specific document. These approaches generally use TF-IDF vectors as the underlying document representation, meaning they are based more on content than semantic similarity. A variety of clustering methods have been researched, with the most effective methods being varieties of hierarchical clustering techniques [18]. Hierarchical clustering groups very similar items together, and then groups similar clusters together repeatedly. This allows for increasing

specificity when used for document clustering. While most clustering methods are hierarchical in nature, some work has been done using spectral clustering of documents [12], and with using more simple clustering methods such as K-means [38]. These methods are often faster to compute, but require users to specify the number of clusters they want, which can degrade the quality of the resulting clusters, and therefore performance in further tasks.

K-Means

K-Means clustering was one of the first widely used approaches for document clustering. This clustering approach is a very simple approach, and has seen widespread use in many various fields of machine learning. This clustering approach requires the user to set a single parameter, K, which indicates the number of clusters that the algorithm should create and assign instances to. In this approach, K centroids are chosen randomly in the dataset. After selecting centroids, a distance between each row and each centroid is computed. Each row is assigned to the cluster corresponding to the centroid it is closest to. After assigning all rows to a cluster, the centroid for each cluster is computed based on the rows in the cluster. After recomputing the centroids, the distances for each row are recomputed and rows are updated if they are closer to a different centroid. This process is repeated until no rows change clusters. This clustering approach has the advantage of being easy to understand, and is relatively efficient to compute. Though this approach works well in many cases, it also requires users to have an understanding of the underlying distribution of the dataset, and makes assumptions as to the number of clusters within the dataset, which may not be accurate. These downsides led to other clustering approaches being developed in an attempt to resolve these issues.

Spectral

Spectral clustering is a clustering technique that focuses on clustering the eigenvectors of the similarity matrix of a set of data points. A similarity matrix is a matrix of N by N where

the value for each entry is the similarity score between two points in the dataset. These matrices are generated by computing pairwise similarity scores for all points in the dataset. This can be costly as the size of the dataset increases, and as a result this technique is less efficient than K-Means clustering. After computing the similarity matrix for the dataset, the eigenvectors for this matrix are computed. These eigenvectors are then used as input to a clustering technique, such as K-Means. For the purposes of document clustering, this is often done by using TF-IDF vector representations for documents, and calculating the pairwise cosine similarity between all vectors.

Hierarchical

Hierarchical clustering techniques are another alternative clustering technique that have been applied to document clustering problems. This family of techniques encompasses a few closely related techniques, each with slight differences. These techniques all share a common advantage over K-Means clustering in that users do not need to specify the number of clusters that exist in the dataset. In general hierarchical techniques cluster group areas of data that are close together, based on some distance metric. Points that are within a threshold distance from each other are considered to be members of the same cluster, and those that are above that distance are considered to be separate clusters. These clusters are then compared, and those that are within some larger threshold are grouped together into a larger cluster. This process is repeated until all data points are part of a single cluster. This method results in a dendrogram of clusters, that contain fewer points and are closer together as one moves to the leaves of the dendrogram. Users can then select a certain point of the dendrogram and consider all branches that exist at that point to be the clusters that exist in the dataset.

2.4 Topic Modeling

Topic modeling is another approach that is used to represent and sometimes group documents for retrieval by search engines and similar systems. These approaches focus on general views of content, and allow documents to contain different topics, which may be distinct, unlike in clustering where documents exist in only one distinct cluster. These methods are more reliant on maintaining the underlying meaning of documents with regards to topic areas, and as such use different techniques to obtain compressed representations for documents. The most commonly used techniques in this field are Latent Semantic Analysis, or LSA, and Latent Dirichlet Allocation, often referred to simply as LDA (not to be confused with Fisher's Linear Discriminant Analysis). In addition to techniques based on these approaches, some other approaches have been used, such as using topological representations of documents such as in Adaptive Topological Tree Structures.

2.4.1 LSA

Latent semantic analysis is one of the oldest techniques used for topic modeling, and was first used by Hofmann [15]. Latent Semantic Analysis (LSA) treats a document as a distribution of topics. The algorithm for performing LSA is as follows. First a term frequency matrix is constructed, where rows are words, and columns consist of documents and the value for each row, column combination is the number of times the word occurs in that document. After generating this term frequency matrix, a singular value decomposition is applied, obtaining three new matrices, one of which is diagonal. A new matrix is then constructed by using the two non-diagonal matrices. This results in an approximation of the original matrix, but with some difference in values. These variations in values reduce the differences in representations of documents from similar topics.

2.4.2 LDA

Latent Dirichlet Allocation is a technique used primarily for topic modeling and was first used by Blei et al [5]. LDA treats documents as weighted mixtures of topics, where each topic is a probability distribution over a vocabulary of words. This differs from LSA which does not view topics as probabilistic mixtures of words, but rather as discrete sets. This means that every word in a document can be attributed to a topic, and that a paper can be viewed as a combination of the topics to which the words in the document belong. LDA is widely used in topic modeling, sometimes with additional supplemental data added. In particular, inclusion of author information has been added by Liu et al to help understand the impact author connections have on topic distributions [23]. In addition, others have added publication date information in an attempt to analyze topic trends and how the distribution of topics changes over time [2]. While LDA is the most common method for representing documents for topic analysis, other techniques exist, most interestingly those that use topological representations, such as Adaptive Topological Tree Structures.

2.4.3 Adaptive Topological Tree Structures

Adaptive Topological Tree Structures (ATTS) are an older method for topic modeling that use a hierarchical system of self-organizing maps to perform topic modeling with the ability to visualize the system and the topic groupings [13]. Self-organizing maps (SOMs) are a simple form of neural networks that are trained in an unsupervised manner to perform dimensionality reduction. Adaptive Topological Tree Structures use a hierarchical tree approach to train a series of SOMs to group documents into topics based on TF-IDF vector representations of documents. Nodes in the tree form children based on an entropy measure based on the documents in each node. In the event a node has an entropy value above a certain threshold, new nodes are created as children of that node and documents are assigned individually to the new nodes such that the entropy value remains below the

threshold. This forms a hierarchical structure that allows for more specificity as one moves down the structure.

2.5 Keyphrase Extraction

The objective of keyphrase extraction is to obtain subsequences (or keyphrases) for documents that accurately represent the content of the document. Most methods for this are trained in an unsupervised manner without a gold standard set of keyphrases that the approach attempts to match or include in its list of keyphrases for a document. Extracted keyphrases allow easy querying of databases for documents that are relevant to a researcher. Methods for keyphrase extraction are primarily graph based in nature with some approaches that are supervised and use more complex machine learning techniques.

2.5.1 Graph Based Approaches

Graph based approaches are a popular technique in the field of keyphrase extraction. In most graph based approaches, a graph is built for each individual document. These approaches have the advantage of being relatively easy to compute, and are also normally unsupervised, meaning a large corpus of labeled training data is not necessary for these methods to be effective at generating keyphrases. The vertices of these graphs are most frequently candidate key phrases. The edges of these graphs are usually filled in based on some metric of similarity between the key phrases. After building a graph a variety of techniques are used to reduce the number of candidate key phrases. A few of the most common graph based approaches are discussed below.

One popular graph based approach to keyphrase extraction is TopicRank [7]. TopicRank creates a graph by clustering phrases into topics, and using the clustered topics as the vertices of a graph, where the edges of the graph are weighted by the semantic relation

between the two topic areas. These vertices are weighted using TextRank, a graph based approach for weighting how important topics are to a document. After weighting using TextRank, the candidate keyphrases are chosen by selecting the keyphrase closest to the centroid for the highest scoring topic nodes in the graph.

In addition to TopicRank, another graph based approach considered is DoCollapse [14]. This approach generates a graph based on potential keyphrases from a document and uses topological data analysis techniques to reduce this set of keyphrases to a much smaller number that can be evaluated for accuracy much quicker. This technique is discussed more in depth in the Topological Data Analysis section of this paper.

Both TopicRank and DoCollapse have been shown to be effective methods for automated keyphrase extraction. Though these graph based approaches are effective methods for keyphrase extraction, there has also been work done to use more complex supervised techniques for automated keyphrase extraction as well.

2.5.2 Supervised Machine Learning Appraches

In addition to the graph based approaches to keyphrase extraction described above, some work has been done using neural networks to obtain keyphrases for documents. These approaches, unlike TopicRank and DoCollapse, are supervised approaches, and as such require a corpus of data with gold standard keyphrases for documents that the model can be trained to predict accurately. The most common of these approaches is an algorithm known as Kea [41]. This algorithm uses the TF-IDF vectors and the position a phrase first occurs in a document as input features to a Bayesian network which attempts to accurately predict whether a phrase is a keyphrase or not. This technique has been found to be effective at generating keyphrases on a number of different datasets.

Though the Kea algorithm is effective, further improvements to it have been made by including more inputs and a more complex machine learning model for predictions. One of these techniques, described by Nguyen and Kan includes additional positional information,

as well as other information about words in the keyphrase [29]. This technique calculates a section vector, which indicates where in the document the keyphrase occurs. In addition to this feature, the authors also include part of speech tagging for words in the keyphrase as well as if it includes acronyms. These additional features are added to the initial features used in the Kea algorithm and used as input to a Bayesian network. These additional features were found to increase performance over the original Kea technique.

2.5.3 Other Methods

In addition to methods using graph-based approaches and neural network based approaches, another common technique for keyphrase extraction is to apply clustering techniques to candidate phrases. These techniques, like graph based approaches, are unsupervised, meaning they work well on datasets without gold standard keyphrases to use as training samples. Two examples of these techniques are TopicalPageRank and KeyCluster. Both of these techniques were found to be effective techniques for keyphrase extraction on a variety of datasets.

TopicalPageRank is a technique that combines LDA and TextRank to generate keyphrases [16]. Technical phrases are used as input to an LDA model, which results in a list of topics. TextRank is run for each topic that may occur in a document to obtain keyphrases for each topic. These are then weighted by the probability of a topic being a part of a document. This technique has also been found to be effective on a variety of different datasets.

KeyCluster is a technique that clusters individual terms from a corpus and uses those clusters to generate keyphrases for a document [24]. This technique removes common stop words from documents, and computes a semantic similarity score for words based on their co-occurrences. These similarity scores are used as input to a variety of clustering techniques, including spectral clustering and hierarchical clustering techniques. After clustering the terms for a document, exemplar terms are chosen from the clusters and used to select noun phrases containing those terms as keyphrases for the document. This technique

has also been shown to be an effective technique on a variety of different datasets.

2.6 Topological Data Analysis

2.6.1 TDA in NLP

Multivariate data can be viewed as points that have a dimensionality equal to the number of features describing them. Due to the large space given this dimensionality, and the fact that there is an underlying process that generates the points, it is highly likely that the space is sparsely populated. As a result, the data points actually exist on a lower-dimensional surface (or manifold). Identifying this manifold allows for exploiting interesting properties of the space, and may allow for more interesting observations.

TDA has been applied to a variety of fields, from image recognition to biological data. The field is still relatively young and as such has not been widely applied in the field of natural language processing. However, there have been some recent attempts to apply TDA to several problems in the NLP domain.. Using TDA on embeddings generated from Word2Vec is a logical application due to the properties of Word2Vec embeddings.

Graph Based Approaches

A number of applications of TDA in the realm of NLP have used graph based approaches in their algorithms. A few examples with a summary of the general technique are given below.

DoCollapse DoCollapse is a technique for keyword extraction developed by Guan, Hui, et al. that takes advantage of the topological structure of documents [28]. The basis of DoCollapse is a semantic graph of candidate keyphrases, where the vertices of the graph are candidate keyphrases, and the edges of the graph connect two vertices if there is suffi-

cient overlap between the words in both keyphrases. A topological collapse is applied to this graph to select the actual keyphrases for a given document. This topological collapse combines two vertices if the content of one vertex is completely contained in another vertex adjacent to it. This process simplifies the graph and reduces the number of edges, and therefore the number of keyphrases.

Legal Judgment Prediction In addition to its use in keyphrase extraction, topological data analysis techniques have also been applied to legal judgment prediction based on written documents describing cases. Documents describing legal cases have an underlying topological structure that is similar to that found in other technical documents such as research papers. In a paper written by Zhong et al, topological data analysis techniques were applied to predict legal judgments based on embeddings of sentences from documents [42]. These techniques were found to outperform conventional TF-IDF based techniques in terms of prediction accuracy. This task can be viewed as similar to topic modeling as the topic of a paper is dependent on the semantic content of the document itself in a similar manner to a judgment being dependent on the semantic meaning of the sentences or facts representing the case.

2.6.2 Mapper

Mapper is an algorithm for performing topological data analysis developed by Singh et al [36]. The algorithm provides an approximation of the simplicial complex for a data set. Mapper has been applied to a variety of datasets and provides a tool for visualizing high dimensional datasets. The process of generating a mapper model for a given dataset is relatively simple, and dependent on a small set of parameters; metric, lens, resolution and gain. A mapper model is generated by first projecting the high-dimensionality data into a lower-dimensional representation by using a lense function. The lense function is intended to reduce the dimensionality of the data space while maintaining interesting properties of

the observed data and their relationships. Common lenses include the first two coordinates of tSNE [25] or UMAP [26] projections. Next, a cover of the lower-dimensional space defined by the lense function is defined. This cover can be thought of as an overlapping set of bins, the union of which covers the entire space defined by the lense function. The number and size of the bins is determined by the resolution parameter. The binning process is also affected by the gain parameter, which determines the amount of overlap between bins in the cover. After binning the low-dimensional data, clustering is performed on the high-dimensionality representation of the data points that fall into each bin. Each of these clusters becomes a point in a mapper model. Points are connected with an edge if the data in one cluster in a bin is also in another bin. An example of a mapper model generated on a sample of points in the shape of a circle can be found below in Figure 2.3.

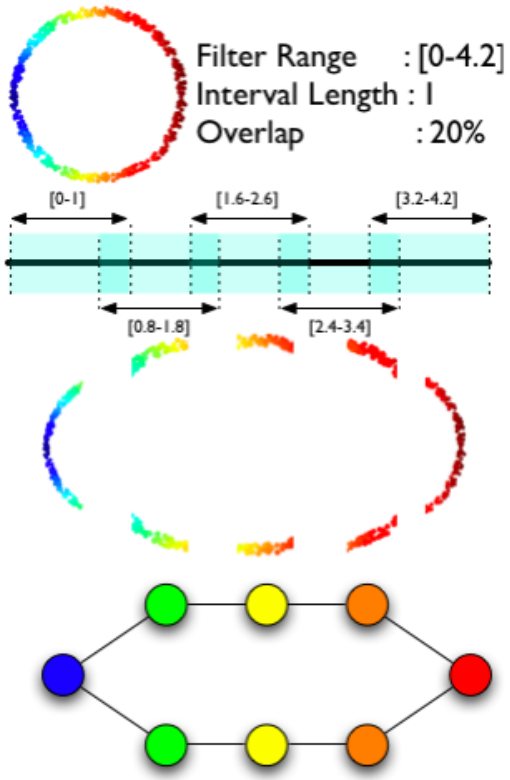

Figure 2.3: A visual example of the process used to create a mapper network from an input point cloud.

Mapper has been used for medical data, visual image recognition, and has been proposed to be effective when applied to NLP tasks. Mapper maintains the underlying topological properties of the initial dataset in many cases, and can also be used to detect clusterings of data that may elude other analysis techniques. Mapper has been expanded into a hierarchical variant called topological hierarchical decompositions (THD) that progressively separate mapper models into smaller subsets.

2.6.3 Topological Hierarchical Decompositions

Topological Hierarchical Decompositions (THDs) were first described by Kramer et al. and used to evaluate risk performance for loans in [8]. In this paper, the authors found that THDs provide the ability to explain how certain features work together to result in different groups of data in a dataset. In addition they emphasized the technique's resilience with regard to noisy data, and its usefulness for visualizing and analyzing large datasets. These features make THDs promising for analyzing text corpora due to their large size and potentially noisy nature.

As with the simpler mapper models, THDs have a relatively small number of parameters necessary for building a successful model. These parameters include the same metric, lens, resolution and gain parameters as in simple mapper models, but also include a resolution increase parameter. The process of generating a THD results in an approximation of a multiscale-mapper as described in [11]. In this approximation, only the resolution parameter is changed, with constant increases as one moves down the construct. The general process for generating a THD is as follows. A mapper model is generated on the entire dataset using a set of starting parameters, commonly a resolution of 1, resulting in all data falling into a single bin, and as such a single node in most cases. After generating an initial mapper model, all data that falls into a connected component that contains a sufficient number of rows is used to generate a new mapper model, with the resolution of the lenses increased by the amount specified by the resolution increase parameter. This process is

repeated on each connected component in each mapper model that has sufficient rows as defined by the user. This results in segmentation of the data into components that are very closely related, and a hierarchical tree structure of models. The following figures show the branching process that occurs in a THD and how the nodes in the displayed THD correspond to individual Mapper models. Figure 2.4 shows the root node of the THD structure on the left, and its corresponding mapper model on the right.

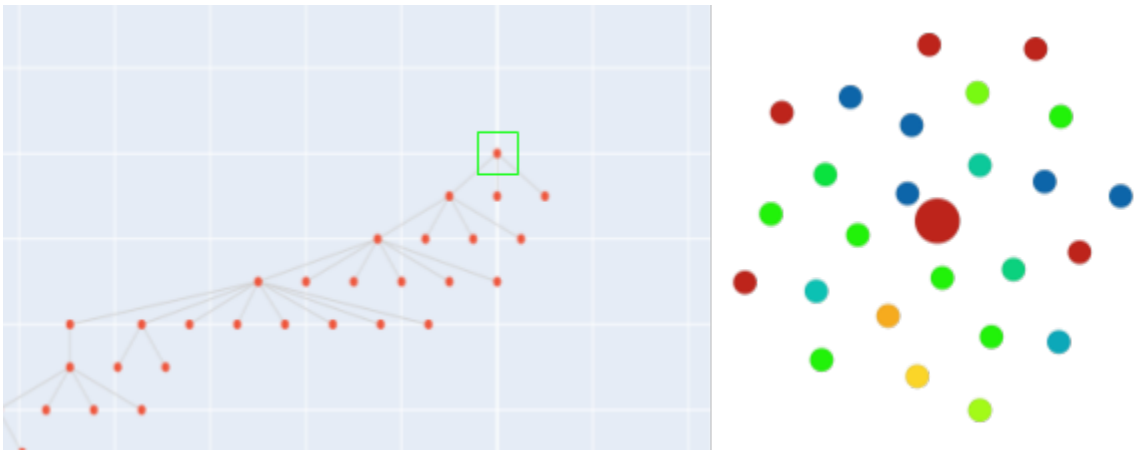

Figure 2.4: A visual representation of the root network of a THD. Node size is indicative of the number of points in each node, indicating one large portion of data and several small outlier nodes that are almost immediately shed.

This model contains one node, as a result of the resolution parameter creating only one bin. The resolution is increased in a constant step until the model begins to separate, as shown in Figure 2.5.

In this figure, the node directly before the branching step is highlighted and its corresponding mapper model is shown. The model still contains one main connected component, but there are very few edges connecting the main portions of it. Further increasing the resolution will cause these edges to be removed and as such form two connected components, as shown in Figure 2.6.

In this figure the branch node is highlighted in the structure, and the mapper model

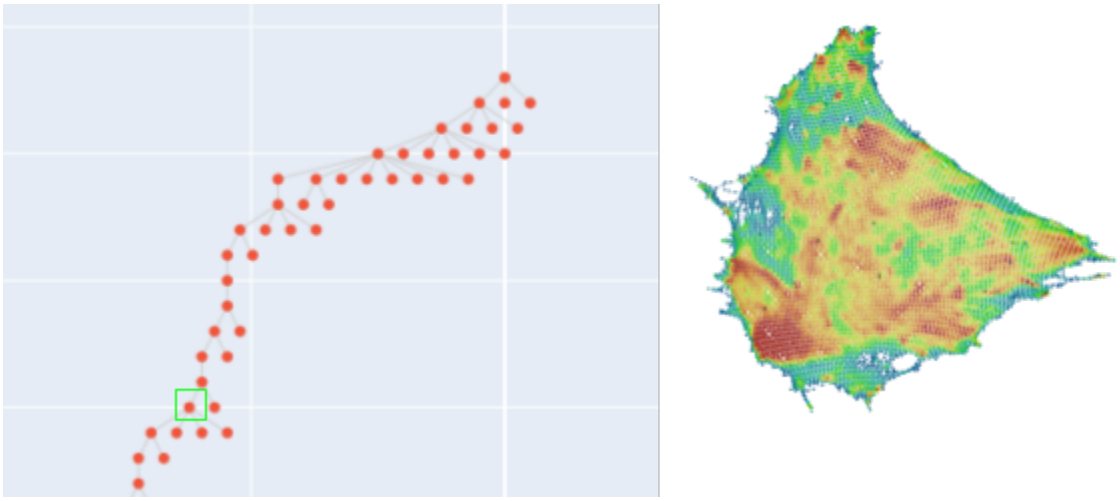

Figure 2.5: The network corresponding to the node immediately before a branch in a THD is shown. The node is highlighted on the left with a green box, and the network is displayed on the right.

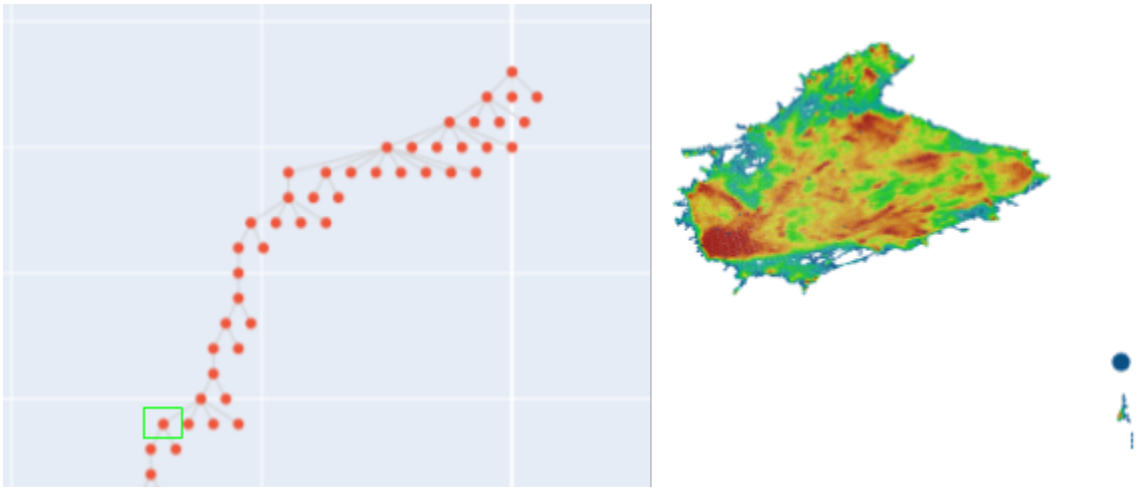

Figure 2.6: The branch node of a THD is shown on the left highlighted by a green box. Its corresponding network is shown on the right. Two individual connected components are visible, which correspond to new branches that will form.

is shown, containing two main connected components, that correspond to the two nodes that are children of the current node. As both of these models are above the user defined threshold for continued segmentation, they will each form a new branch in the THD and be segmented separately.

Figure 2.7 shows the left child of the branch node and its corresponding mapper model which corresponds to the large connected component in the branch model. The model in

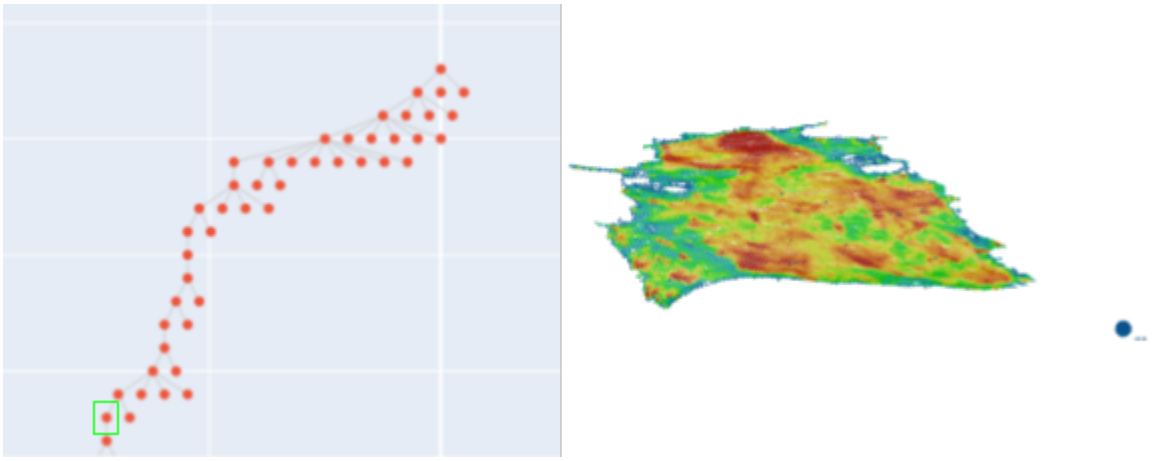

Figure 2.7: The left child of the branch node in Figure 2.6 is highlighted with a green box on the left. Its corresponding network is shown to the right.

Figure 2.7 consists of one large connected component, with a small number of outlier points that will not be further segmented. The main component shows the formation of another flare which will likely form another branch relatively quickly with similar properties to the right branch generated in the current step shown below.

Figure 2.8 shows the right child of the branch node, which corresponds to the small connected component in the branch model.

In Figure 2.8, the model consists of many extremely small connected components with very few points in each component. This is the result of a relatively large resolution being applied to a small subset of data, which occurs frequently during THDs and indicates that the points in this branch are closely related and difficult to separate until a very large

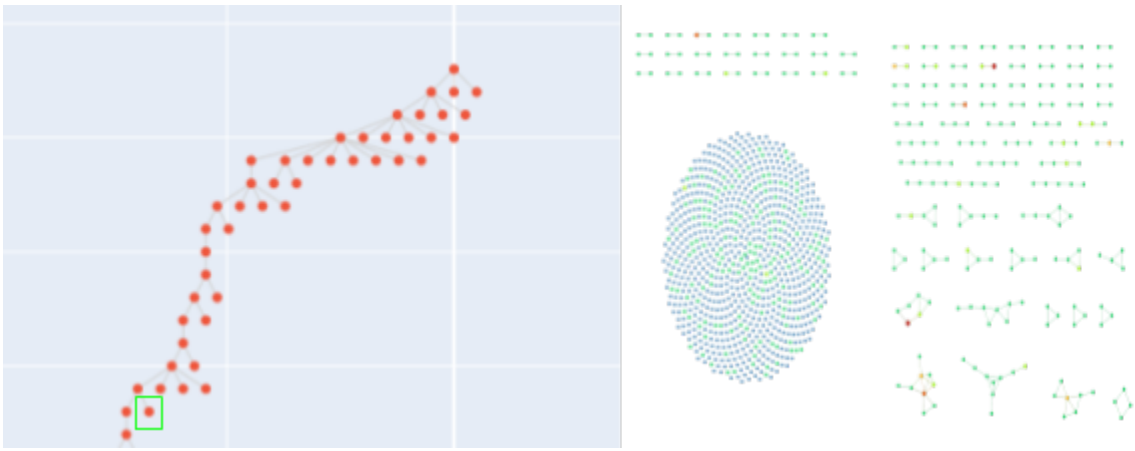

Figure 2.8: The right child of the branch node in Figure 2.6 is highlighted with a green box on the left. Its corresponding network is shown to the right.

resolution is reached.

The resolution for the branch containing the larger connected component is increased until another branch occurs or the maximum resolution is reached. The branch corresponding to the smaller component does not progress further due to the fact that all components in the resulting model are below the threshold set to continue segmenting.

The resulting tree structure of a THD can easily be queried by data both included in the features used when constructing a THD, or by additional features not used when constructing the THD. These queries show which nodes of a THD specific data points occur in. This allows detection of underlying trends in the data that may not be obvious when initially considering features to use when processing a dataset. In addition to querying a THD by outside features, one can also visualize where specific rows exist in a THD, as well as use nearest neighbors based approaches to determine where additional data would likely fall into a THD. These approaches use a K-nearest neighbors voting algorithm to determine which nodes a data point would be in based on the full dimensional features. As THDs only segment groups of sufficiently large data, outliers are often left unsegmented, and as such THDs are not negatively affected by such outliers. Though these are often left unsegmented, it is possible to view when and where these outliers are removed from the model, allowing for analysis of why they are outliers, especially in applications where

outliers may be of particular interest, such as anomaly detection.

As mentioned before, the properties of THDs make them promising for a variety of NLP related tasks. For example, THDs constructed on the sentences of documents may group them effectively into groups of topics which may be used to perform topic modeling and as input for keyphrase extraction methods. In addition, THDs can be used to generate document-level embeddings that may be used as alternative representations for document clustering techniques. As such, this paper explores the usability for THDs in both document clustering and topic modeling tasks for scientific documents.

Methods

3.1 Dataset Generation

The dataset used for this thesis was synthesized from two different sources. The majority of the dataset was sourced from the COVID-19 Open Research Dataset Challenge on 06/01/2020 [39]. This dataset was supplemented with a large number of closely related abstracts obtained from the Microsoft Academic Graph by running a query for papers related to coronaviruses, angiotensin and a small sample of related words, as described in the MAG section below.

3.1.1 CORD-19

The COVID-19 Open Research Dataset Challenge (CORD-19) is a corpus of approximately 140,000 research articles related to coronaviruses, with over 60,000 of the articles containing full-text [39]. The articles in the corpu sare primarily related to SARS-Cov2, though there are a significant number of papers relating to previous coronavirus outbreaks, and a smaller number related to other types of coronaviruses, that are less related to SARS-Cov2. For the purposes of this project, only articles containing a full-text paper were used, and articles lacking full-text were discarded. This selection of documents resulted in approximately 50,000 documents being kept. This number was decreased after further processing removed documents. Though the number of papers is substantial, it was necessary

to obtain more documents that were also related to coronaviruses to supplement this data in order to ensure there was sufficient data to be used to train a word embedding model effectively.

3.1.2 MAG

In order to supplement the data obtained from the CORD-19 challenge, abstracts related to coronaviruses, angiotensin, and ACE-2 were obtained from the Microsoft Academic Graph. The Microsoft Academic Graph (MAG) is a database of scholarly articles maintained by Microsoft [37]. The database allows querying by keywords, topics, and other metadata. A query on documents with keywords related to coronaviruses, angiotensin-ii, and a small list of synonyms and related words was submitted to obtain a large number of abstracts that could be used to supplement the CORD-19 documents. This query resulted in 156172 abstracts being returned. Though this number is much larger than the documents obtained from the CORD-19 dataset, the length of documents is shorter, which is acceptable for the purposes of this project, due to the fact that the content of the sentences remains similar. These abstracts were then processed in combination with the CORD-19 documents to build a complete dataset.

3.1.3 Merging Corpora

Initially, all documents from both corpora were checked for overlap between the two sets. In the event that a document existed in both the full-text dataset and the abstract only dataset, the abstract was removed from the abstract dataset. This helped to prevent duplication of sentences and slightly decreased the number of sentences that need to be processed. After checking for duplicate documents, each document was verified to be written in English by using the Python langdetect library. After these processing steps, the merged corpus consisted of 48409 full-text documents, and approximately 100,000 abstracts. This

corpus was then further processed in preparation for use with a fastText word embedding model.

In the event of overlapping papers between the two primary corpora, the full-text version was kept and the additional abstract was discarded. The resulting corpus consisted of 48409 full-text documents, and 57118 abstracts.

3.1.4 Preprocessing

After generating the initial dataset from the two subsets of data, and removing documents not written in English, the documents were further preprocessed in multiple steps to ensure that the data was properly formatted for use with the algorithms used. The first step after removing all non-English papers was to convert each of the remaining documents into a list of sentences. This was achieved by using the pretrained Natural Language Toolkit (NLTK) Punkt sentence tokenizer [19]. This tokenizer uses an unsupervised approach to find sentence boundaries. The full body of each document is passed to this tokenizer, and a list of sentences is obtained as a result. This list of sentences is then further processed into tokens. Each sentence obtained from the sentence tokenizer is then tokenized using a regular expression tokenizer. This converts each sentence into a list of individual tokens, which are usually single words, but can include words combined with hyphens. This list of tokens is further processed to remove stop words and the remaining tokens are stemmed. For stop word removal, all tokens are converted to lowercase and compared to the list of English language stop words as defined in NLTK. This process can remove all words in a sentence, and when this occurs the corresponding sentence is removed from the dataset, as it is impossible to generate an accurate embedding for it. After removing all stop words and converting the tokens to lowercase, they are stemmed using the WordNet lemmatizer found in NLTK. This lemmatizer converts all words from a common stem to the common stem word, for example; good, better, and best would all be converted to the word good. Lemmatization allows for a simpler vocabulary, which in turn improves word embedding

performance. After all preprocessing is performed, the resulting dataset consists of over 7 million of tokenized sentences. These tokenized sentences are used to generate a vocabulary of words and to train a fastText word embedding model.

3.1.5 Vocabulary Generation

After converting all documents to lists of tokenized sentences, the unique words in the dataset are compiled into a vocabulary. This is done by iterating over all documents and getting the unique words in each document, along with the number of times each word occurs in the document. These word-count pairs are added to the vocabulary if they don't already exist, and in the event they do, the number of occurrences is added to the stored value. In addition to the number of total occurrences of a word in the entire corpus, the number of distinct documents that a word occurs in is also tracked in the vocabulary. These values are later used to select a subset of the vocabulary that is used to generate a smaller subset of the entire corpus, that was used to construct the final resulting THD models.

3.1.6 Metadata Dictionary Generation

In addition to generating a vocabulary of unique words and tokenized representations for each sentence in a document, a secondary dataset of metadata for all documents in the dataset was generated in parallel. This dataset consists of unique identifiers for all documents as well as a set of metadata associated with each document. This dataset allows for efficient query of sentences based on the metadata, such as authors, title, and publication venue and date, for the documents they are found in, allowing for effective visualization, discussed later in Section 3.5.1.

After processing all documents fully, the resulting list of tokenized sentences was used to train a fastText word embedding model from scratch. This process used for training the model is described below in the following section.

3.2 Word Embedding Generation

The sentence embeddings used in this work were generated by averaging word embeddings. Generating sentence embeddings by aggregating word embeddings has been shown to be an effective method for many tasks, as shown by Weiting et al. The technique used in that paper is based on Word2Vec embeddings, and the primary deficiency noted by the authors is in sentiment classification tasks. As a result of the above findings, a fastText model was chosen to generate word embeddings. This type of model has similar advantages to Word2Vec models in that it has a small enough number of free parameters to be trained on a relatively small corpus of specialized data, matching the format of the corpus used in this paper. In addition to being a relatively simple model, fastText also performs better in tasks most tasks, especially those requiring subword information as opposed to Word2Vec based approaches. This improved performance should allow for sentence embeddings generated in a similar method to that used in the paper by Weiting et al. to perform similarly or better on most tasks, with potentially improved performance in sentiment classification tasks as well. While other approaches such as GloVe and BERT based word embedding models chave demonstrated better performance on several downstream NLP tasks, these approaches require using either a pretrained model or a much larger corpus. Due to the specialized nature of the corpus in this paper it was determined that more general pretrained models would likely perform worse, and that the size of the corpus is insufficient to train either of these more complex models. Though SciBERT and bioBERT are both trained on scientific data, these models are both trained on more general scientific and biological data, and as such would still require further fine-tuning which would be difficult with the size of the corpus used.

Two separate fastText word embedding models were built and trained on the full dataset. Both fastText models used a CBOW architecture for the underlying model, and were trained using similar parameters. The first model was trained to generate embeddings with 100 dimensions, and was trained for 5 epochs with a window size of 5 words.

The second fastText model was trained to generate word embeddings with 300 dimensions, and was also trained for 5 epochs with a window size of 5 words. This secondary model was trained as Adi et al. had found that using word embeddings with 300 dimensions produced from Word2Vec outperformed those generated with 100 dimensions.

The parameters chosen for the fastText model were derived from the findings of Adi et al. which explored the performance of different embedding dimensions on a variety of sentence embedding models. The authors of this paper found that using a Word2Vec model, using 300 dimensional embeddings provided a small improvement in performance on all tasks as opposed to the default 100 dimensional embeddings [1]. Further increases of dimensionality were found to decrease performance in some tasks, with no noticeable improvement in others. Due to the similarity in architecture between Word2Vec and fastText models, 300 dimensional embeddings were used when training the model. In addition to embedding dimensionality, there are three other important parameters for training fastText models; the underlying architecture, window size, and the number of epochs to train the model for. The fastText model used in this paper is based on a CBOW model. This type of architecture was chosen as it is faster to train, and the performance improvements provided by skipgram based models are relatively minor and not as necessary for the purposes of this paper. The window size for the model trained in this paper was set to 5, as this is found to provide adequate performance across all tasks. This means the model looks at 5 words at a time when learning the embedding for a word, and due to the CBOW architecture chosen, two preceding words and two following words are used to predict the word in the center of the window, which is the embedding being learned. Finally the number of training epochs was set to 5. This is the default as recommended by the creators of fastText. Due to the relatively small size of the corpus this value was selected in an effort to avoid overfitting the model which would degrade the quality of the resulting word embeddings.

After selecting parameters, the fastText model was trained on the entire corpus of tokenized sentences, though it was set to only generate word embeddings for those words

that occur more than 5 times in the corpus. This decision was made to ensure that there were sufficient examples of all words receiving embeddings to ensure that the embeddings were accurate. This also prevents words with slight misspellings that occur very rarely from receiving a word embedding as this would likely result in a greater dissimilarity than from using subword information on a fully trained model to obtain their embeddings. The small relative size of the corpus and the simplicity of the embedding model chosen allowed for the training process on the entire corpus to be completed in less than two hours on a stock Ryzen 3900X CPU with 32 GB of RAM.

3.3 Sentence Embedding Generation

As described earlier, the sentence embeddings used in this paper are generated by averaging word embeddings from a fastText model trained from scratch solely on the documents in the corpus used in this paper. The decision to generate sentence embeddings based on the average of word embeddings was motivated by three factors. First, Weiting et al found that the embeddings produced in this manner, though relatively simple, are effective at a variety of tasks [40]. Second, similarly to generating word embeddings, the small size of the corpus used in this dataset prevents training more complex sentence embedding models such as those based on BERT. Third, the relative simplicity of the embeddings generated by averaging word embeddings allows for easier understanding of the separations that occur when the embeddings are used in a THD. This will allow for easier analysis and visualization by end users, whereas embeddings from more complex models are less suited to this task. The following process was used to generate sentence embeddings from the fully trained fastText model.

First, a list of word embeddings is generated for each tokenized sentence. This list is generated by using each token as a query to the trained fastText model, which returns the corresponding embedding. In the event a word is not in the vocabulary used to train

the model, the subword information for the query is used to find a new embedding for this word. This situation primarily occurs due to small misspelling errors, though very rare words that occur less than 5 times in the corpus would also be embedded this manner. After generating a list of word embeddings, the embedding vectors are summed, and the resulting vector is divided by the length of the input list of tokens. This results in a single embedding for each sentence with the same dimensionality as a single word. This identical dimensionality is useful as it allows for easy comparisons of sentences by to input sequences by converting input words to their corresponding embeddings and using similarity measures to find results.

The procedure described above was used to generate two datasets of sentence embeddings, one with 100 dimensions, and a second with 300 dimensions. These two datasets were compared to determine whether the dimensionality of the embedding space had a noticeable effect on the quality of further results. The embeddings generated by this method are then used as the input dataset for a THD to be constructed on.

3.4 THD Construction

As described above, THDs are an extension of the mapper algorithm that provides an approximation of multiscale-mapper in the case where the resolution is increased consistently. A THD provides useful properties for understanding and visualizing the way that sentence embeddings group together. In general the end nodes of a THD contain items that are very close together in the feature space used to construct the THD.. This means that elements falling within the same leaf node of a THD tend to be more closely related (in terms of their features) than those found in other THD leaf nodes. With respect to sentence embeddings, end nodes of a THD contain sentences that have similar meaning in their full 300-dimensional embedding space, which can be viewed in some cases as similar topics. Nodes that are not directly connected contain sentences that are much less similar and can

be viewed as separate topics. As one moves up the tree structure towards the root, the content of the nodes becomes less specific and can be viewed as more general topic areas. These features make THDs a logical choice to use for topic modeling as one can easily view the centroid of each group which can be viewed as the topic. In addition one can query THDs by external data for sentences similar to an input, allowing for viewing of topic trends with regard to authors, time or even journals.

In addition to providing the ability to perform topic modeling, THDs can also be used to generate an embedding for entire documents by tracking which nodes the sentences of a document occur in to obtain a new vector representation of the topics in the paper. These vectors can be combined to create a single vector representation for a document that can then be used with conventional document clustering techniques, or as an input to yet another THD used to cluster documents. This is explored further in Section 3.5.1.

Other methods exist to find similar sentences based on the similarity of embeddings, generally by finding the N highest scoring sentences. These techniques can be effective at finding very similar sentences, but require knowledge of the relative size of the corpus, as well as determining a good similarity cutoff. In addition these techniques are not as effective for finding sentences with similar ideas but differing content. This requires a larger number of sentences to be analyzed, with more focus on lower scoring sentences. Again this technique is affected by the size of the corpus and as such requires the user to know more about it. In comparison, the structure of a THD is affected much less by the size of a corpus, allowing for less knowledgeable users to be effective. In addition, using the nodes of a THD removes the requirement for setting the number of sentences to return or a similarity threshold, as the nodes group sentences into moderately sized groups that can easily be analyzed.

Due to time and space constraints it was necessary to reduce the total number of sentences in the corpus prior to THD construction. . This downsampling was accomplished by removing sentences that did not contain at least one word that occurred in more than 5

documents but less than 10 times in any. Because the combined corpus contained both full documents and abstracts, the length of documents varied substantially. This selection method helped to ensure documents were not completely excluded nor oversampled, while providing a sufficiently small dataset. In the end, the downsampling resulted in a dataset of 788096 sentences. All abstracts were retained, with an average retained sentence count of 1.7. The majority of full text documents were kept, with an average retained sentence count of 12.4. While this preprocessing step successfully decreased the dataset size, it could have potentially removed extremely specialized documents from the dataset, and potential improvements for this step are discussed later.

Several different THDs were constructed from the individual sentence embedding datasets. Only one THD was constructed from the 100 dimensional embeddings. The 300-dimensional embeddings were used to construct a variety of THDs to determine the effects of various parameter choices on the resulting embeddings. Table 1 below shows the parameters used for each of the THDs constructed.

A number of experiments were conducted with a variety of metric and lens choices to explore the parameter space of the THD models and determine the best model parameters.These parameters were explored as they have the greatest impact on the resulting structure of a THD. Lens options included two based on common dimensionality reduction techniques, Neighborhood lenses, and UMAP lenses. The Neighborhood lenses provide an approximation of two components obtained by running tSNE on a dataset. Similarly, UMAP lenses provide an approximation of the first two components obtained by running UMAP on a dataset. Neighborhood lenses were chosen due to the fact that these lenses have been found to be effective when used to construct THDs on other types of data []. UMAP lenses were also chosen due to their similar function to Neighborhood lenses, but have been found to have improved performance when compared to Neighborhood lenses.

Two different metrics were explored, cosine similarity and correlation. The cosine similarity metric computes a pairwise cosine similarity between all sentences in the dataset

and was chosen as it has been found to perform well for other document similarity related tasks in the past. Equation 3.1 shows the equation used to calculate cosine similarity between two vectors.

$$Cosine(X,Y) = 1 - \frac{\sum_{i=1}^{N} X_i Y_i}{\sqrt{\sum_{i=1}^{N} X_i^2}\sqrt{\sum_{i=1}^{N} Y_i^2}} \tag{3.1}$$

The correlation metric is based on the Pearson correlation coefficient to calculate pairwise distances. This metric was chosen as it often performs well on a variety of tasks that THDs have been applied to, whereas a cosine metric was chosen based on its performance in other NLP tasks. The equation for computing this value for two input vectors is given in Equation 3.2.

$$Correlation(X,Y) = 1 - \frac{N\sum_{i=1}^{N} X_i Y_i - \sum_{i=1}^{N} X_i \sum_{i=1}^{N} Y_i}{\sqrt{N\sum_{i=1}^{N} X_i^2 - \left(\sum_{i=1}^{N} X_i\right)^2}\sqrt{N\sum_{i=1}^{N} Y_i^2 - \left(\sum_{i=1}^{N} Y_i\right)^2}} \tag{3.2}$$

n addition to metric and lens parameter exploration, the effects of the resolution correction parameter were also explored. This parameter regulates the resolution increase at points where a branch occurs in a THD. It is expected that when using this parameter, a THD will contain longer branches and as a result may have better segmentation into groups. This parameter was explored as some THDs constructed using it have had structures with better segmentation. Table 3.1 below shows a summary of the combinations of these parameters explored when constructing THDs.

Sentence Embedding Dimensions	Metric	Lenses	Resolution Correction
100	Cosine	Neighborhood	No
300	Cosine	Neighborhood	No
300	Cosine	Neighborhood	Yes
300	Correlation	Neighborhood	No
300	Cosine	UMAP	No

Table 3.1: Table of THD parameter settings explored.

In addition to the parameters explored above, the following parameters remained constant across all THDs constructed. These parameters include the starting resolution, set at 1. This is the most common starting resolution as it ensures that all data exists in a single starting node for the THD, preventing any initial loss of data. The gain parameter, which determines the percentage of overlap between filter bins was set to 1.5. This is a relatively low gain value for a THD, and was chosen as a lower gain value helps to speed up segmentation. Higher values for gain result in increased overlap, and thus more connectivity between nodes in a model, requiring higher resolutions, and thus more time to separate. The resolution increase parameter, which determines how much the resolution for each lens increases at each step, was set to 10. This value is within the normal range of values for this parameter, and was chosen to ensure that the resolution did not increase too quickly and result in poor segmentation due to the low gain value, while remaining high enough to ensure quick segmentation.

As the dataset used for these THDs consisted of a large amount of unlabeled documents with no ground truth, it was difficult to quantitatively validate the effectiveness of the resulting THDs for clustering and similarity searches. As such, a tool was built to enable qualitative analysis and visualization of the resulting THD structures and the underlying groups in them.

3.5 THD Visualization Tool

A visualization and analysis dashboard was built to help assess the resulting sentence embeddings and the separation achieved by individual THDs. The goal of this dashboard was to provide the ability to find groups of similar sentences in a THD, and to allow for searching for sentences similar to an input sequence. In addition to finding similar sentences, the dashboard was also designed so that similar papers could be discovered based on where their sentences occurred in a THD. These features would allow for easy analysis of the quality of THDs for document and sentence similarity tasks. This dashboard consists of a variety of modules that allow users to query the underlying THD model using a variety of methods. In addition to enabling querying of THD structures, the dashboard also provides visualization functionality for the global THD structure, as well as the ability to visualize the data included in each individual node of a THD.

3.5.1 THD Queries

The first goal that the dashboard was designed to achieve was to provide the ability to quickly and efficiently query a THD to find sentences that may be relevant to a user. This was achieved by building an efficient method to query the THD based on a variety of different techniques, described below. The methods for querying the THD can be divided into two main groups, similarity-based searches, and metadata-based queries.

Similarity Search Queries

The first main group of querying techniques is the group of searches based on similarity. These methods allow a user to input a series of query words and return the sentences in the dataset that are most similar to the input sequence. This is done in three distinct methods. Two of these methods can be viewed as providing boolean search functionality based on the input sequence.

First, users provide a series of at least one input word, and the dashboard will return the sentences in the dataset that contain at least one of the provided input words. These sentences will be highlighted in the THD and individual node visualization modules that are described further below. In addition to highlighting these sentences in the THD structure, the dashboard also displays the sentences with the highest percentage of words that were in the input sequence, along with the percentage itself. This search method is similar to the boolean OR function, in that a sentence only needs to contain at least one word from the input sequence to be returned as a result.

The second search method that provides boolean search capability is one that allows users to input a sequence of words, and returns sentences containing all of the words in the input sequence. Similar to the first method described above, this search method highlights the resulting sentences in both of the main THD visualization modules. This search method also returns the sentences containing the highest percentage of words from the input query, as well as their corresponding percentages. This search method can be viewed as analogous to the boolean AND function, as all words from the input query must be in a sentence in order for it to be returned by the search algorithm.

The third similarity-based search differs from the previous two in that it uses the actual embeddings that the THD was constructed from to provide search results to the user. In this search method, the user again provides a sequence of at least one input word, and the algorithm will highlight the relevant sentences in the visualization modules. Relevant sentences for this search technique are found by using a cosine similarity metric. First, the embeddings for the words provided in the input query are obtained from the fastText model used to generate sentence embeddings. These embeddings are obtained by either direct lookup in the case that the word was already in the dataset the model was trained on, or by approximation based on subword information if the word was not in the initial vocabulary of words. After obtaining the embeddings for all words in the query, they are averaged to obtain a single embedding, in an identical manner to the averaging to generate

sentence embeddings. This resulting embedding, viewed as a pseudo-sentence embedding is then used to find the most similar sentences based on their sentence embeddings. This is done by computing pairwise cosine similarity between the pseudo-sentence embedding and all other embeddings in the dataset. After calculating the similarity scores, the top scoring sentences are highlighted as in other methods and returned to the user with their corresponding similarity scores.

In addition to providing the highest scoring sentences and the ability to visualize where they occur in a THD, the dashboard also provides the ability to view statistics about the metadata for the resulting sentences, and the papers that they occur in. The documents that contain the most sentences found by the query are analysed, and distributions of metadata for those papers can be displayed to users to allow for further analysis. Users can see the titles of the documents that contained the most sentences, as well as the distributions for the authors of those documents and the journals they were published in. These are provided in the form of automatically generated pie charts for a quick visual analysis by the user. In addition to author and journal information, when available for the documents a histogram of the publication year for the documents is also generated and provided for the user when desired.

The sentence search methods described above allow for users to find sentences that may be similar to an input sentence, and visualize the distribution of those sentences across the THD and the documents in the dataset themselves. The metadata visualization module also allows for users to quickly find documents that contain similar content and provides potential new inputs that users can use for metadata-based searches also provided in the dashboard.

Metadata Queries

The second main method for querying the THD provided by the dashboard is by document metadata. This querying module allows users to find documents and sentences that match

the criteria of a series of metadata queries based on authors, journals and publication date.

Users can query the dataset for sentences from documents that were published by an author from a series of provided authors. This search functions by searching through all documents in the original dataset, and finds all papers that have at least one author from an input list of authors. After finding all papers that meet these requirements, the sentences that correspond to these papers are returned using the unique ID for each document returned by the query. These sentences are highlighted in both of the main THD visualization modules using a similar method to the similarity-based queries described before.

Users can also query the dataset for documents that were published in one of a series of provided journal venues. This search functions similarly to the author based queries in that the original dataset is iterated over, and the documents that were published in one of the journals provided by the user are returned. The sentences for these documents are highlighted in both THD visualization modules in an identical method used to highlight those found by author queries.

In addition to querying by authors and journal publications, users can also query documents that were published within a year range. This search functions similarly to the previous two queries, except looking for documents that were published within the year range provided by the user. The sentences from these documents are highlighted in both visualization modules using the same methodology used for both author and journal queries.

The individual metadata-based queries described above can also be used in combination to provide further restriction for the documents and sentences returned and displayed. Users can select any combination of two individual methods, or can select all three to perform a query. These combined queries are performed in a similar method to individual queries, but instead check all requirements at once, providing for an efficient search based on all requirements that takes approximately the same time as any individual query.

These metadata queries allow users to query THDs to find documents that may be similar, based on the metadata from them. This functionality can be combined with similarity

based searches as described above to enable users to quickly find documents that may be relevant to an input query.

3.5.2 THD Visualization

In addition to providing the ability to query the dataset for sentences that are similar to an input query or meet metadata criteria, a second goal for the dashboard was to provide a method for visualizing and qualitatively analysing a sentence embedding-based THD. This was obtained by creating two additional visualization modules that allow users to visualize the overall structure of a THD, as well as the content of individual groups of the THD individually.

THD Structure Visualization

The first THD visualization module is a display that allows users to see the overall structure of a THD, and select individual nodes to further analyse. This is done by creating a tree-based representation where each node of the tree represents a group in the THD. This tree-based representation is labeled and displayed to the user so they can view how the structure branches and assess quality. This tree can be highlighted by queries run by the user described above. The tree will be highlighted such that a node that is highlighted contains at least one sentence returned by the query. This allows users to view topic areas in the THD, and find groups to analyse more in-depth. Additionally, users can select individual nodes of this tree to restrict searches and visualize the internal content of a group using the individual network visualization module described below. An example visualization of a THD structure is found below in Figure 3.1.

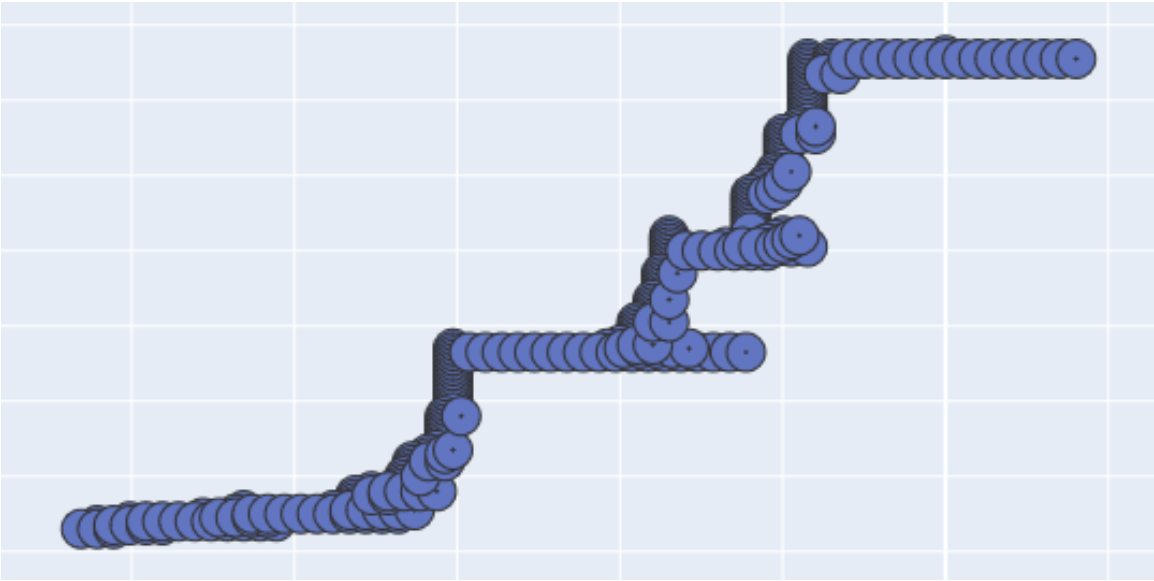

Figure 3.1: Example output of the THD structure visualization. Each node represents an individual Mapper network. Moving down the tree results in more specific topic areas and moving left to right increases the difference between topics.

Individual Network Visualization

The individual network visualization module allows users to visualize the content of individual groups of a sentence embedding THD. This module uses UMAP to generate a 2-dimensional representation for all sentence embeddings from a selected group in the THD [26]. UMAP was chosen for dimensionality reduction due to its effectiveness for handling large datasets, and quality of maintaining structure. These embeddings are plotted, with the corresponding text for the sentence visible when the user hovers over a data point in the plot. This allows users to assess the quality of individual groups of a THD to ensure they contain similar sentences. In addition, these points will be highlighted by any queries run by the user, allowing for visualization of the sentences returned and further analysis of lower scoring sentences found by the query that were not initially displayed to the user. An example of a plot of sentences from a THD group is found below in Figure 3.2.

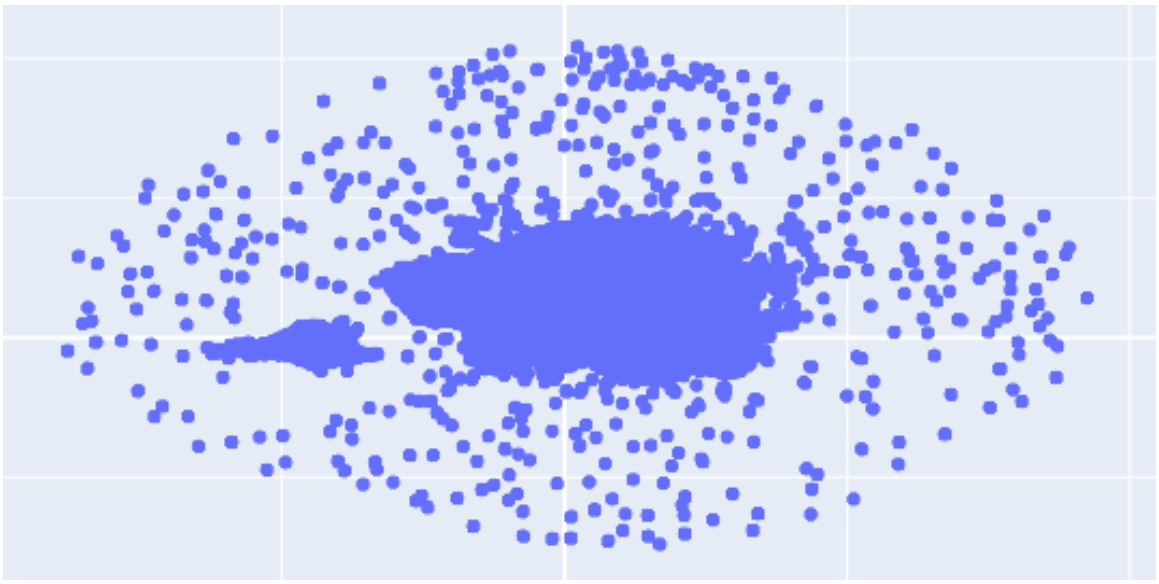

Figure 3.2: Example output of the THD network visualization. Each node represents a unique sentence and can be hovered over to read the content of the sentence. Sentences are embedded in 100 dimensions and then reduced to 2 dimensions by UMAP.

3.6 Document Embeddings

The THDs generated based on sentences can be used to generate document embeddings, which can in turn be used for document retrieval and clustering techniques as an alternative embedding. For the purposes of this paper, this method was evaluated by comparing the clusters generated using embeddings derived from sentence THDs and TF-IDF vector representations. In order to generate these embeddings, a vector representation for the groups of the THD that sentences occur in is generated. This vector representation is a binary vector, with as many entries as there are groups in a THD. The values of this vector are 1 if the sentence exists in a group, and 0 if not. A vector is generated for each sentence in the initial dataset, which can then be combined to generate a representation for an entire document. These vectors are combined by averaging all vectors for the sentences of a single document. As the groups of the THD can be viewed as compressed representations for the topics covered by documents in the corpus, the resulting vectors show how important each topic is to a document. These embeddings can be used to generate document clusters,

but can also be used to perform document similarity searches, by finding the most similar vectors to an input vector.

3.6.1 Document Clustering

The document embeddings generated from the resulting THDs were used as input for K-Means clustering of the documents in the combined corpus. K-Means models using a Euclidean distance metric were chosen for simplicity. For comparison the TF-IDF vector representations for these documents were used as input to another K-Means clustering model using identical parameters. The value of K was selected empirically by exploring values ranging from 3 to 11 with 100-dimensional sentence embeddings. The value yielding the best performance for these embeddings (K=6) was used for all other experiments. Parameters remained the same to ensure that cluster quality was only impacted by the vectors used as input to the clustering model. These two clustering results were compared to determine the effectiveness of the new embeddings for document clustering approaches. The cluster quality was evaluated using the Silhouette coefficient [35] to score how effectively the clusters were separated and how closely together the items in each individual cluster were grouped. This evaluation metric was chosen as it is an effective method of evaluating cluster quality when ground truth labels are not available. Silhouette scores range from -1 to 1 where a higher score indicates a better clustering performance, and a score near zero indicates that clusters are overlapping.

Document Similarity Search

In addition to being used to generate document clusters, the document embeddings obtained from THDs were also used as input for a simple document similarity search. These document embeddings were compared to the same TF-IDF vector representations used for the document clustering comparison. The document similarity search was based on cosine

similarity, and returns the documents closest to an input document vector. The returned documents for identical searches using both methods were compared qualitatively to determine if document embeddings generated from THDs were more effective for document similarity searches.

Results

Figure 4.1 shows a two-dimensional UMAP projection of the 100-dimensional sentence embeddings obtained by calculating the mean vector of the Fasttext embedding for each word in the sentence as described in Section 3.3. This plot shows the dataset consists of what appears to be one main cluster of sentences, with a smaller, tightly grouped cluster of sentences. In addition to these two main clusters, there are a relatively large number of additional sentences that appear to be outliers, with some very small clusters as well. These sentences appear to be related to sentences referencing other documents or licenses for documents and software. These sentences being outliers indicates that the resulting sentence embeddings do capture meaning of the majority of the sentences, and that these sentences are vastly different from the majority of the sentences in the dataset, which is to be expected.

A UMAP projection of the 300-dimensional sentence embeddings is shown in Figure 4.2. This plot consists of a single main cluster with some flairs near its edges. In addition, there are a moderate number of outlier points with a few smaller, dense clusters. This indicates that the increased dimensionality of the embeddings did not adversely affect the ability of the embeddings to capture the general trends of the sentences in the dataset. Similar to the plot of the 100-dimensional embeddings most outliers are again sentences containing references to other authors and licensing terms.

The main cluster of sentences visible in Figure 4.2 consists of a large number of

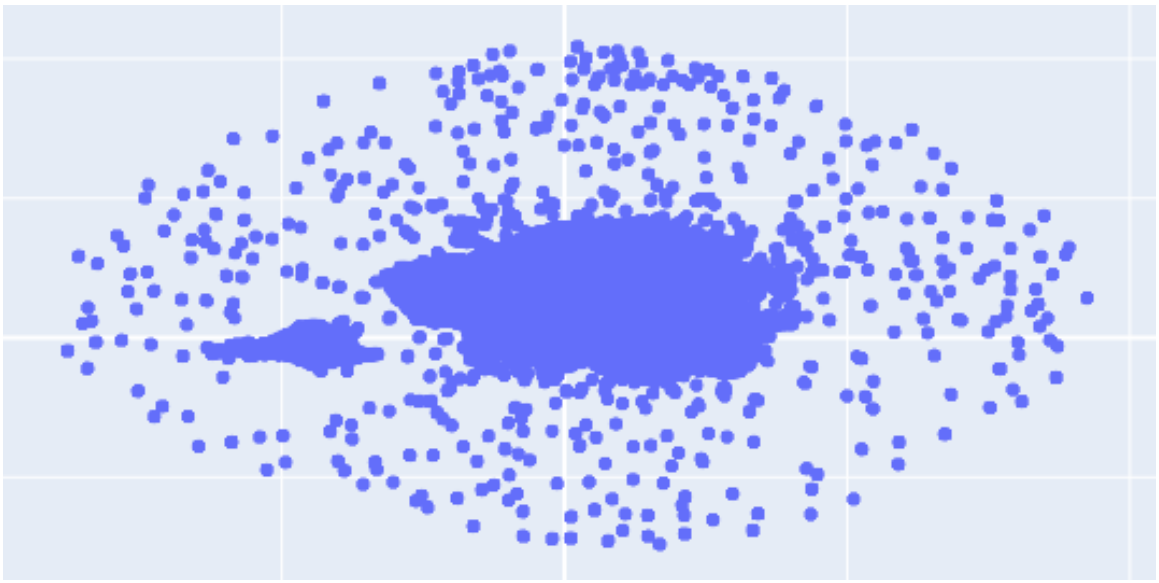

Figure 4.1: Plot of all 788096 sentences embedded in 100 dimensions reduced to 2 by UMAP.

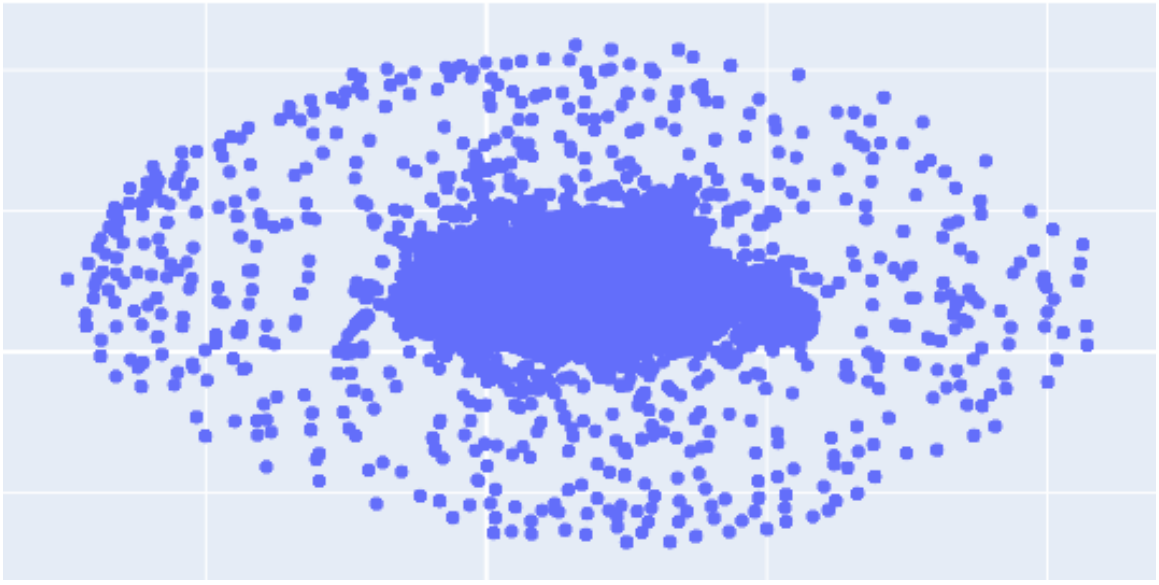

Figure 4.2: Plot of all 788096 sentences embedded in 300 dimensions reduced to 2 by UMAP.

sentences covering varied topics similar to the main cluster in Figure 4.1. This main cluster consists of more dense sub-clusters than were found in Figure 4.1, indicating the additional dimensions allowed for these sentences to group into more identifiable clusters. Figure 4.3. Shows an example of one of these smaller subclusters from the main group.

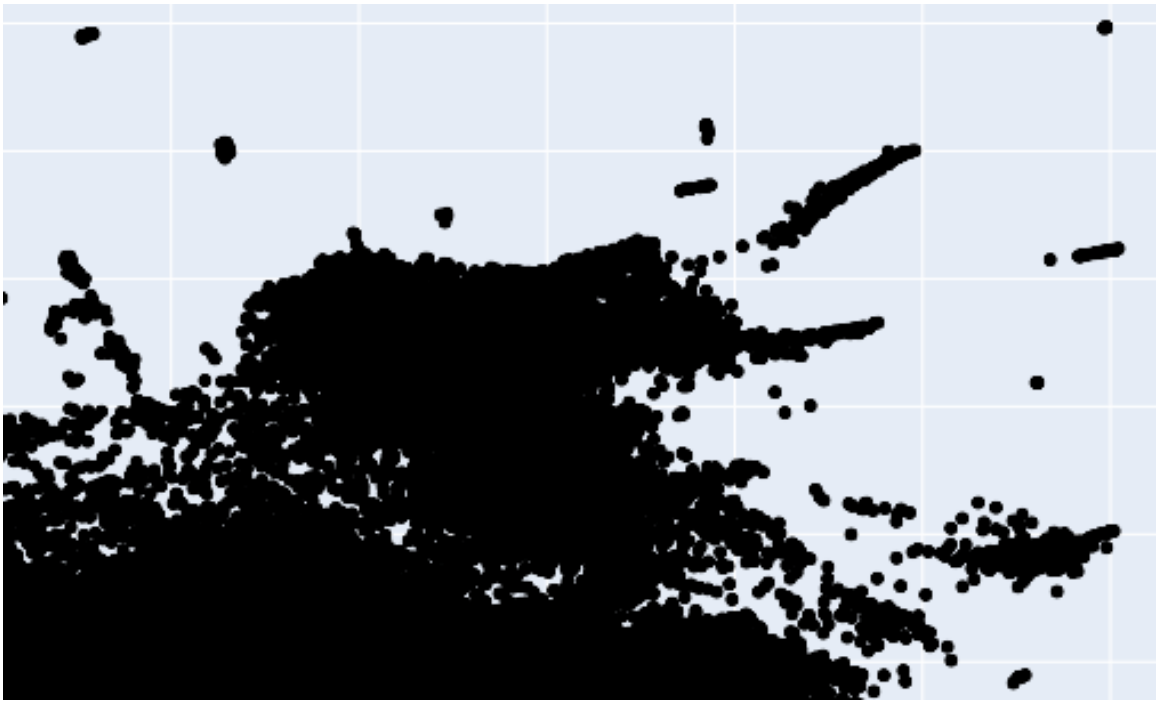

Figure 4.3: Close-up of the upper right region of the main cluster from Figure 4.2, showing tighter grouping into subclusters.

Figure 4.4 shows a cluster of sentences regarding treatment of tuberculosis using isoniazid. Two sentences from this group contain very similar content, indicating that close groupings in the UMAP projection indicate sentences that contain similar content.

This is further reinforced by analyzing a few outlier clusters, shown in Figure 4.5. The upper cluster contains sentences regarding licensing for papers published by Wiley. The highlighted cluster on the right contains sentences discussing usage of papers found on medRxiv.

Figure 4.6 shows an example of two sentences with vastly different content are found

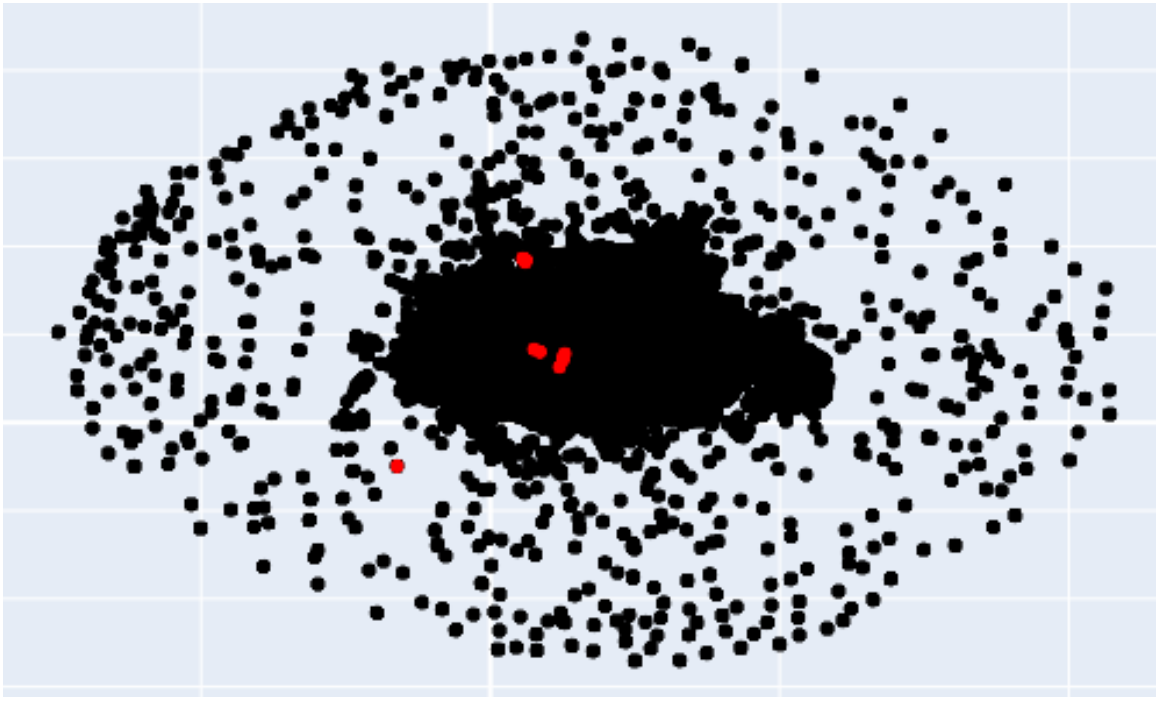

Figure 4.4: Sentences regarding treatment of tuberculosis with isoniazid are found grouped close together. These sentences are found highlighted red, with other sentences colored black.

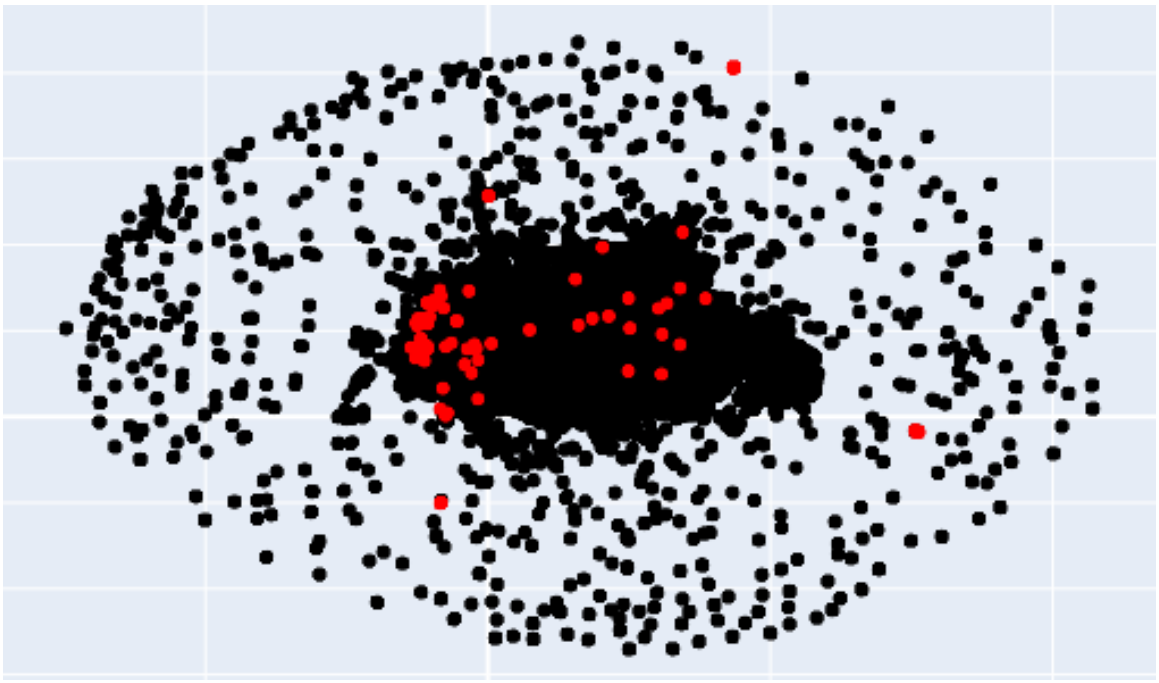

Figure 4.5: Sentences discussing solely licensing and publishing terms are found as outlier clusters in the top and right sides.

separated by a large amount in the UMAP projection. In this figure, the upper sentence discusses antibiotic resistant strains of Tuberculosis, and is located somewhat close to sentences describing treatment of tuberculosis. The lower sentence describes the content of a paper analyzing codon usage patterns of tuberculosis. Though these two sentences both mention tuberculosis, the actual content of the sentences is vastly different, reinforcing the separation shown in the plot.

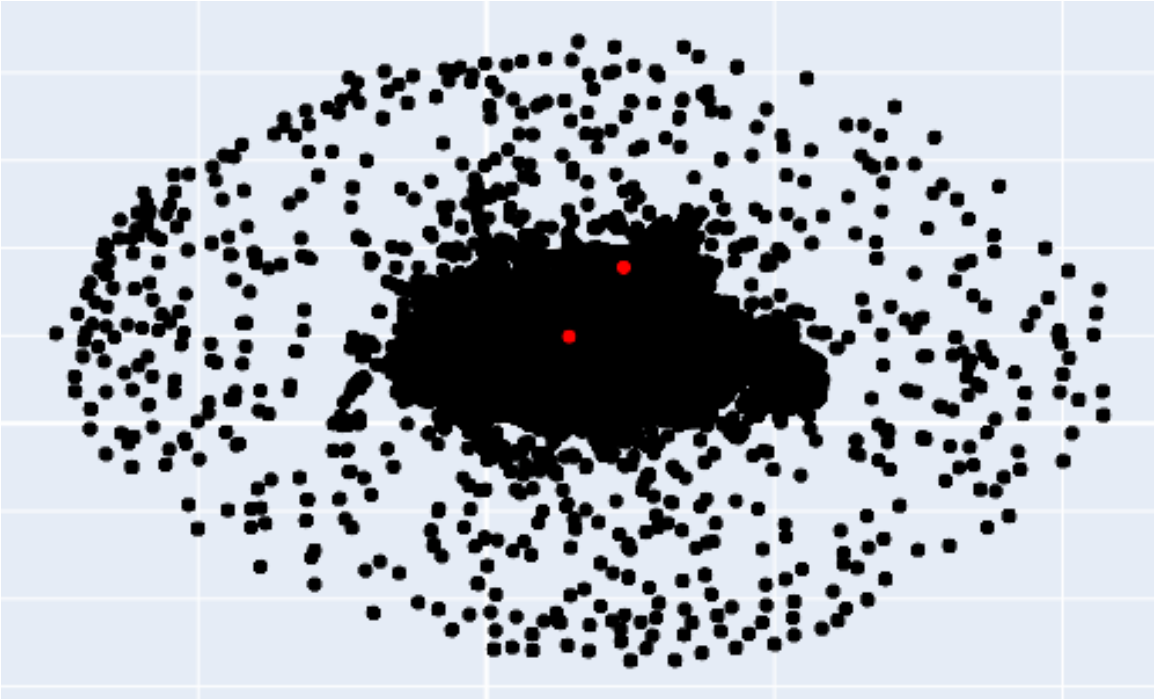

Figure 4.6: Two sentences containing different content are found separated in the UMAP projection. Upper Sentence: For example, the structure of the phylogenetic tree in relation to drug resistance, constructed for M. tuberculosis strains isolated in Russia, suggested that resistance to fluoroquinolones and pyrazinamide was acquired during infection rather than pre-existing in the infecting strain; this in turn suggested that strains resistant to these antibiotics might be less transmissible than susceptible strains. Lower Sentence: In this paper, the codon usage patterns of 12 Mycobacterium tuberculosis genomes, such as the ENC-plot, the A 3 /(A 3 + T 3) versus G 3 /(G 3 + C 3) plot, the relationship GC 12 versus GC 3 , the RSCU of overall/separated genomes, the relationship between CBI and the equalization of ENC, and the relationship between protein length and GC content (GC 3S and GC 12), and their phylogenetic relationship are all analyzed.

In order to assess the quality of the sentence embeddings generated in comparison to current techniques, the distance between the embeddings for two very similar sentences are

compared to the distances between the embeddings for the same sentences generated by a BERT transformer model. These results are shown below in Table 4.1.

Sentence	Averaged Word Embedding Neighbor Index	BERT Embedding Neighbor Index
'It has been known that the SARS coronavirus utilizes the angiotensin-converting enzyme 2 (ACE2) receptor to enter the cell [12] .'	0	0
'It is now known that SARS-CoV-2 utilizes the same receptor of SARS-CoV, which is angiotensin converting enzyme II (ACE2) for viral entry into the host cells [5, 40] .'	1	3
'Akin to its relative SARS-CoV, SARS-CoV-2 uses Angiotensin converting enzyme)-2 (ACE2) as a viral receptor to enter host cells, [15] [16] [17] and ACE2 is an important regulator of intestinal inflammation.'	2	2
'Similar to the SARS-CoV virus implicated in the 2003 SARS outbreak, SARS-CoV-2 facilitates cell entry by attaching to angiotensin converting enzyme 2 (ACE2) located on the cell surface [1] .',	3	1
'15 Other researchers had earlier found that nCoV-2019 utilizes a particular receptor (angiotensin converting enzyme II receptor; ACE2) to gain entry into the cells.'	4	4

Table 4.1: Sentences within a sample dataset and their nearest neighbor index to the first input sentence in the top row of the table.

As shown in the table, the embeddings generated by the technique described above appear to be ordered in a more logical manner than the embeddings generated by a pretrained BERT model. This indicates the embeddings are effective for this very specialized corpus. In addition, all sentences in the sample dataset were found in the same node of constructed THDs, indicating that the THD is grouping similar sentences together in a meaningful way, whereas these likely would not be grouped together otherwise.

The resulting THD structure for the THD constructed from the 100-dimensional sentence embeddings can be found below in Figure 4.7. This figure, obtained from the visualization dashboard described above shows an abstract representation of the branches of

the resulting THD structure. As shown in the figure, the resulting structure contains one main trunk branch, with a series of shorter and smaller branches that may indicate more specialized topics.

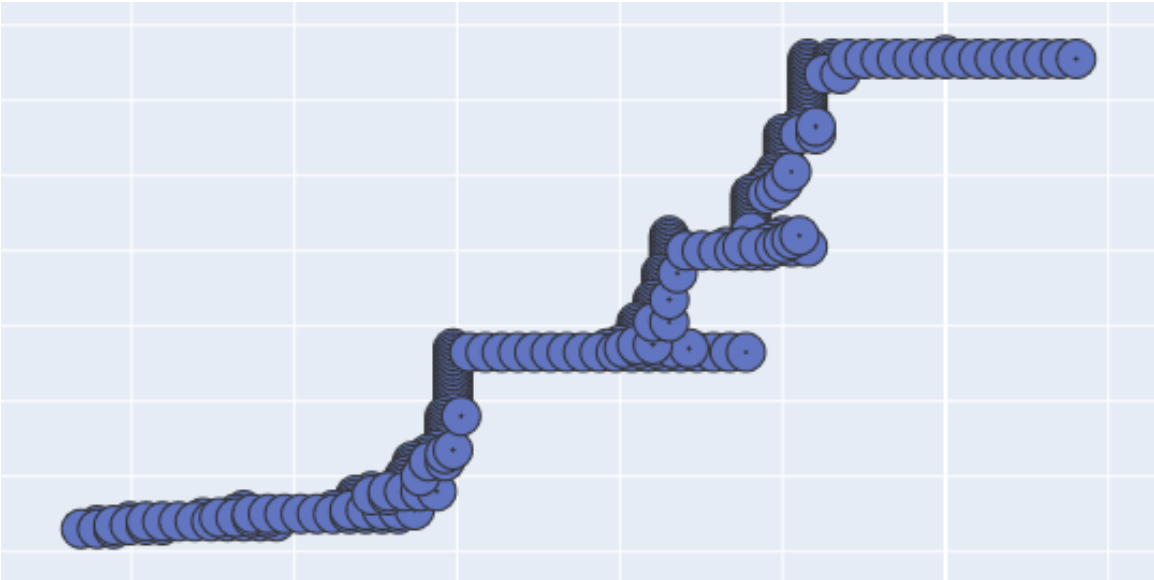

Figure 4.7: Structure of THD constructed using 100-dimensional sentence embedding inputs and cosine metric.

Figure 4.8 below shows the branches of this THD with sentences containing the word coronavirus. Nodes that contain at least one of these sentences are colored green, and nodes without any returned sentences are colored orange.

As expected, these sentences are spread over a large portion of the THD, similarly to their distribution over all sentences in the dataset. Figure 4.9 shows a UMAP representation of all sentences in the dataset, with all sentences containing the word coronavirus highlighted in orange.

This figure shows that the dataset is mainly contained in one large group, with minimal initial separation into smaller clusters. This supports the initial findings of the THD search.

The THD structure for the THD constructed from the dataset consisting of 300-dimensional sentence embeddings using a cosine metric without resolution correction can be found be-

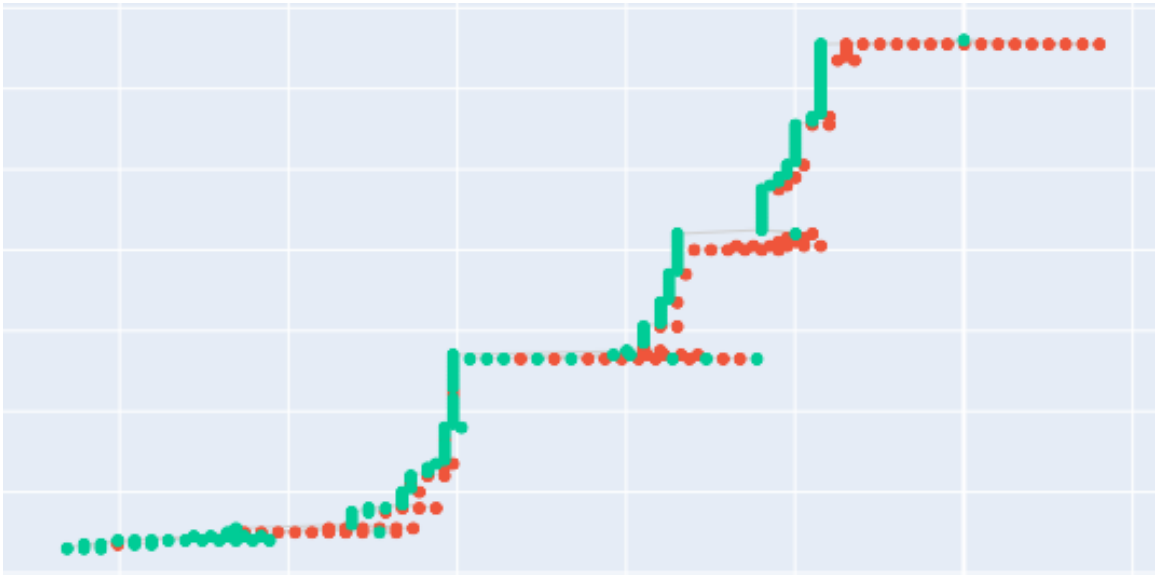

Figure 4.8: THD constructed from 100-dimensional sentence embeddings using a cosine metric with nodes with sentences containing “coronavirus” in green

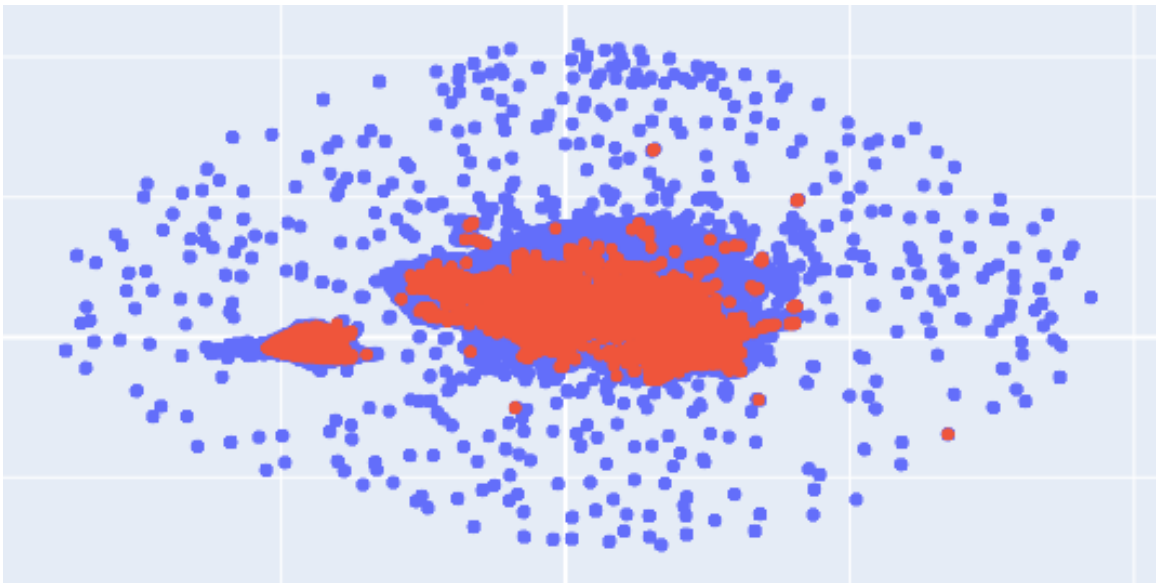

Figure 4.9: All 100-dimensional sentence embeddings plotted with UMAP with sentences containing “coronavirus” in orange.

low in Figure 4.10. As with Figure 4.7, this figure was obtained from the dashboard built for this project. This figure shows that the THD constructed from 300-dimensional embeddings has a similar branching structure to the THD constructed from 100-dimensional embeddings. This THD differs in that some of the initial branching steps include more branches at a level than in the prior THD. These indicate more data was captured that allowed for easier separation with the increased dimensions.

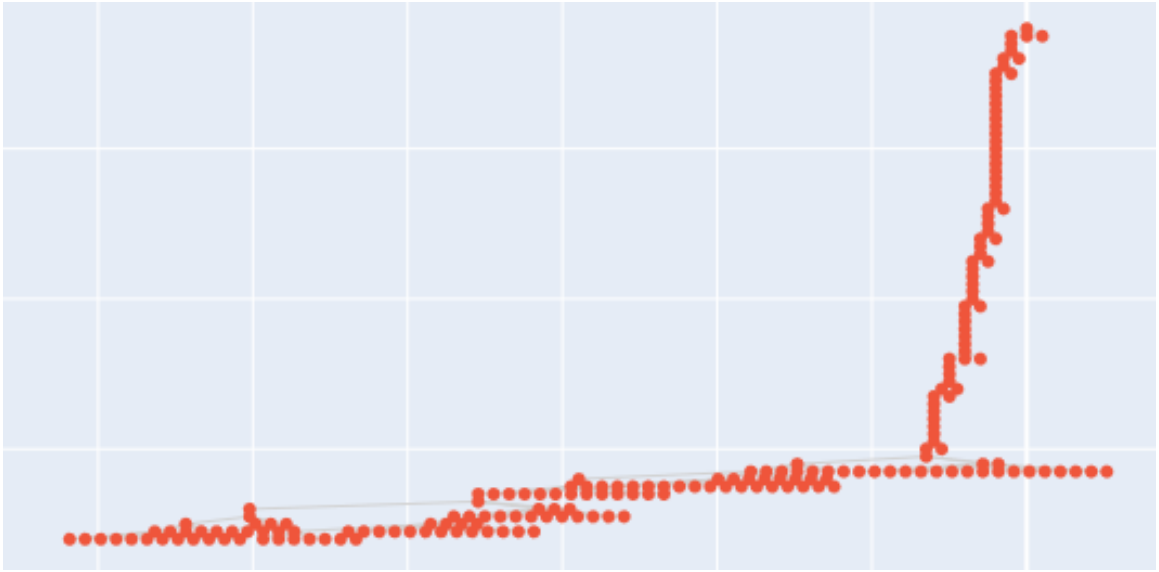

Figure 4.10: THD constructed from 300-dimensional sentence embeddings and using a cosine metric.

Figures 4.11 through 4.15 show the existence of a drug treatment branch in the THD constructed using a cosine metric with UMAP lenses. This branch is highlighted initially in Figure 4.11, with further analysis based on various drugs shown in the following figures.

Figure 4.12 shows the nodes that contain sentences with the word Bupropion in them. These sentences fall into a single leaf node below the node identified as belonging to the drug treatment branch. The sentences in this node discuss the usage of Bupropion as a treatment for tuberculosis, reinforcing that the branch is a general branch discussing various

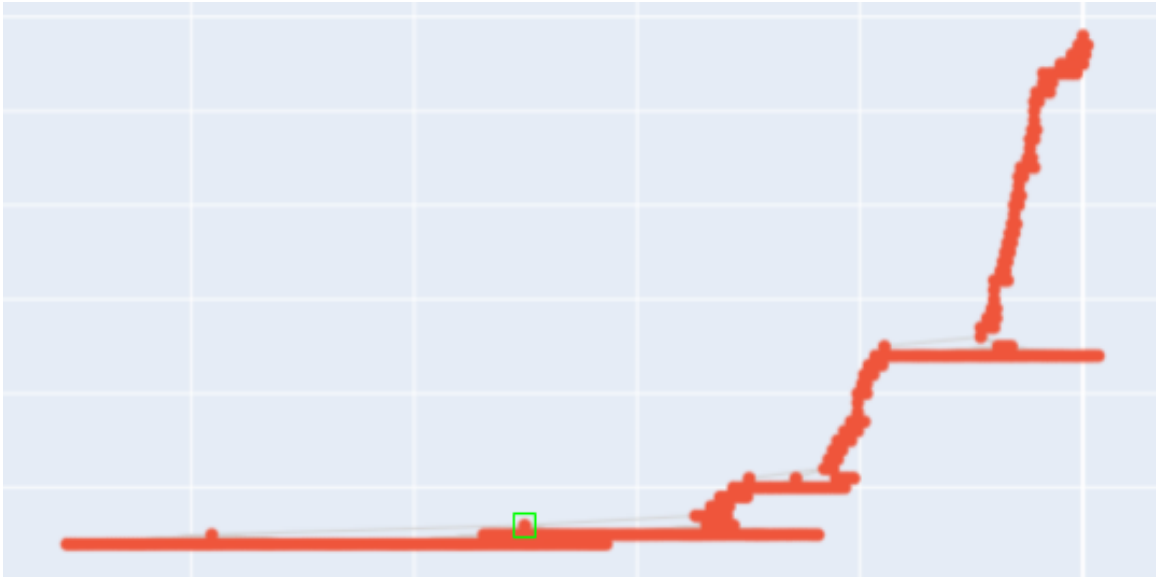

Figure 4.11: THD constructed from 300-dimensional sentence embeddings with a green box indicating the node that corresponds to the immediate parent of a discovered drug treatment branch.

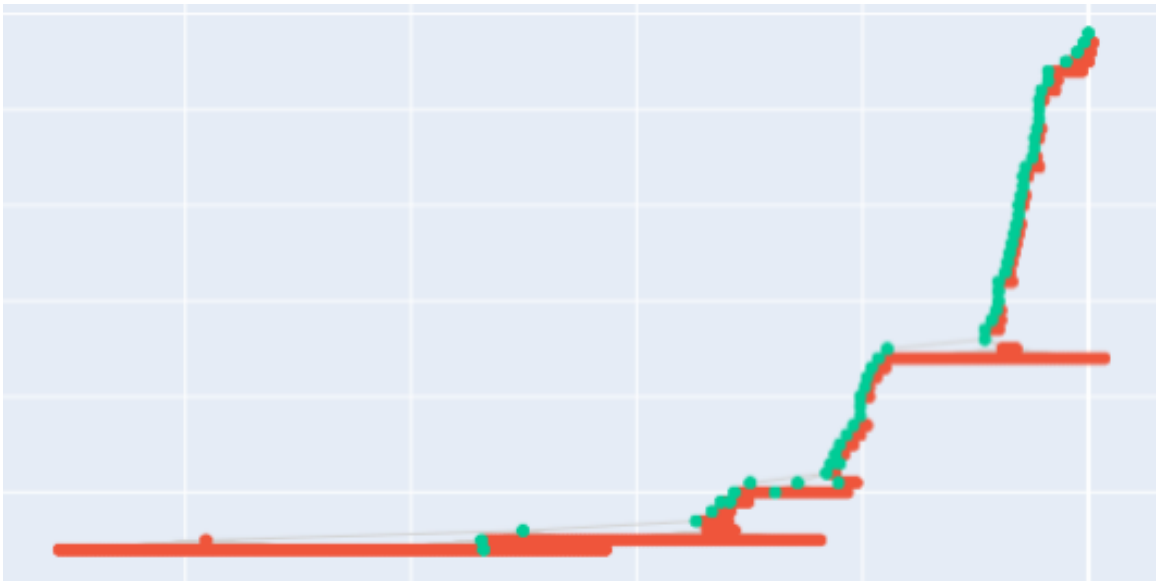

Figure 4.12: THD constructed from 300-dimensional sentence embeddings with nodes containing sentences with the word Bupropion in them highlighted green.

drugs and their usage in treating diseases.

Figure 4.13 shows the nodes that contain sentences with the word Remdesivir in them. These sentences fall into a series of leaf nodes, with a majority in leaf nodes at a level lower than the branch shown in Figure 4.12. This indicates that Remdesivir is discussed in two distinct areas for treatment. Further analysis shows that the sentences in the lowest level nodes discuss the usage of Remdesivir for treatment of coronaviruses, while the sentences in the higher level leaf nodes do not. This indicates that the lower level branch likely contains sentences regarding potential drug treatments for coronaviruses, a more specific subset of drug treatments.

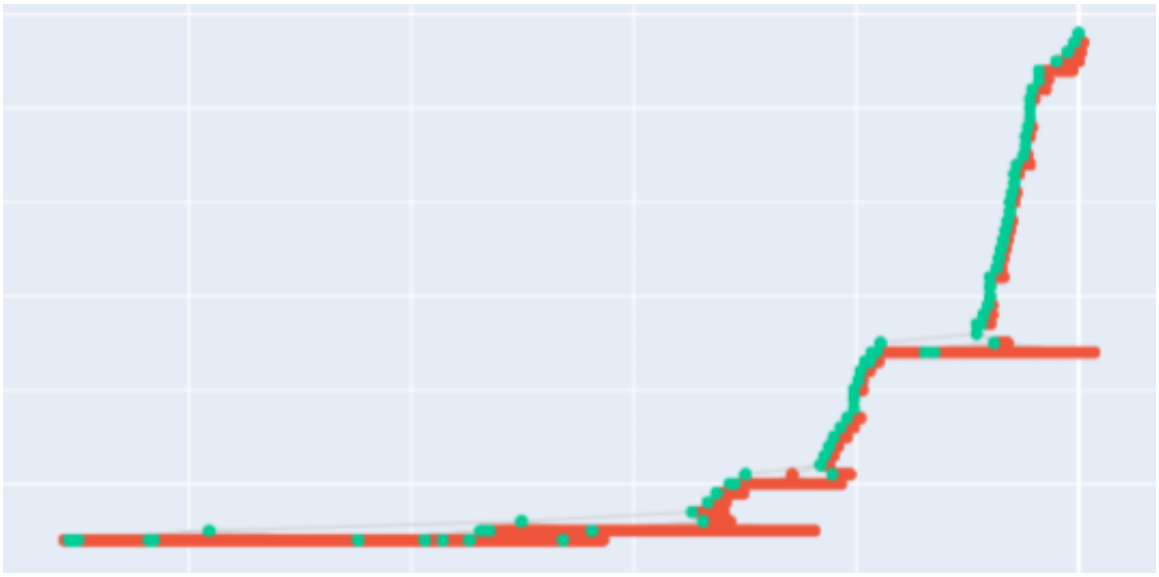

Figure 4.13: THD constructed from 300-dimensional sentence embeddings with nodes containing sentences with the word Remdesivir in them highlighted green.

Figure 4.14 shows the nodes that contain sentences with the word Hydroxychloroquine in them. Similarly to the sentences containing Remdesivir, these sentences again fall into a relatively large number of leaf nodes, at two distinct levels of the THD. Again, the sentences in the leaf nodes at the lowest level contain sentences discussing the usage of Remdesivir as a potential treatment for coronaviruses. This further reinforces the understanding that the parent of these leaf nodes is a grouping of sentences related to drug

treatments for coronaviruses.

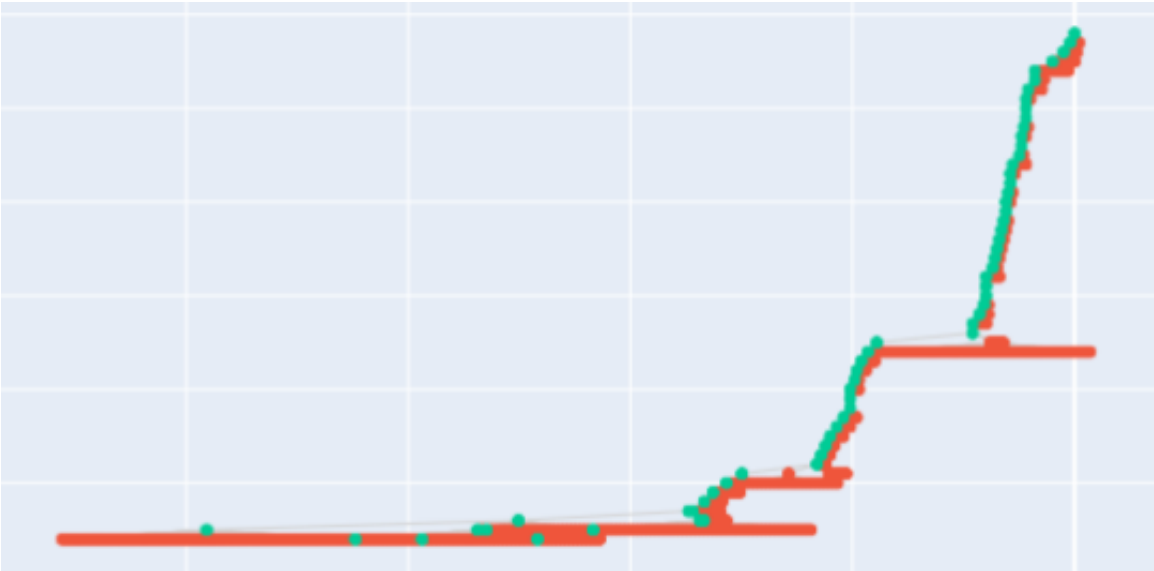

Figure 4.14: THD constructed from 300-dimensional sentence embeddings with nodes containing sentences with the word Hydroxychloroquine in them highlighted green.

Lastly, Figure 4.15 show the nodes that contain sentences with the word Azithromycin in them. These sentences fall into a similar group of nodes as the sentences containing Remdesivir and Hydroxychloroquine. The sentences in the lowest level leaf nodes again discuss the usage of Azithromycin as a treatment for coronaviruses, with the sentences in the remaining leaf nodes discussing its usage for the treatment of other diseases.

Figures 4.11 through 4.15 clearly show the existence of a branch in the THD that captures the topic of drug treatments for diseases, with a lower level branch specifically covering the topic of potential drug treatments for coronaviruses. These groups were quickly uncovered and validated through a small number of queries, enabling close analysis of a small subset of sentences with quickly, and without any initial knowledge of the relative quantity of these sentences in the corpus. These groups show the ability for THDs to group similar sentences together at varying levels of abstraction.

The sentences found in leaf nodes of this THD were compared to the nearest sentences returned when using a clustering approach to determine the effect the THD had on grouping

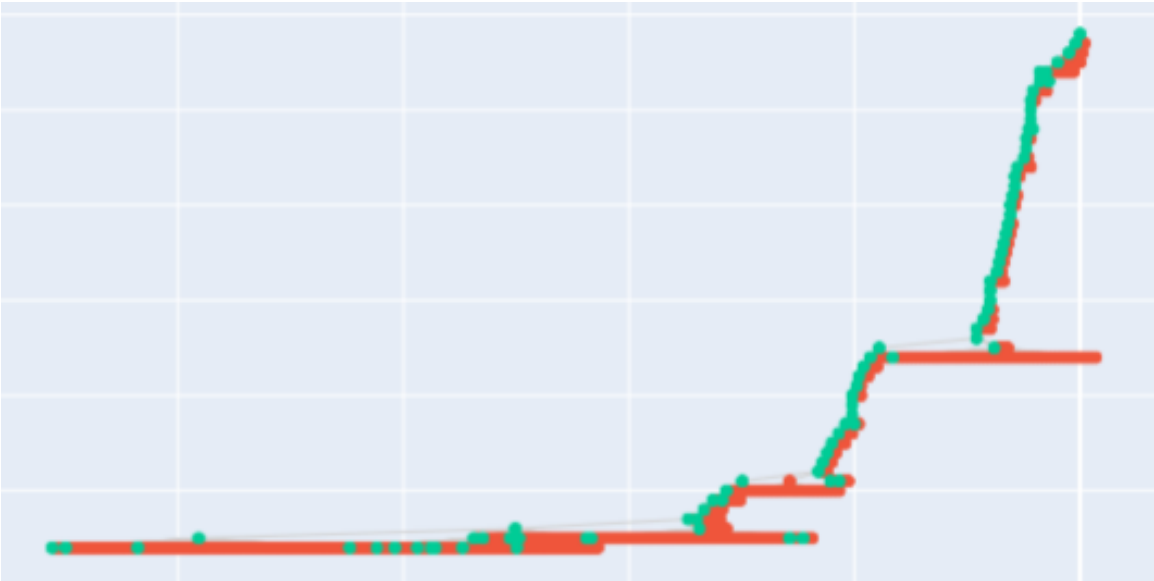

Figure 4.15: THD constructed from 300-dimensional sentence embeddings with nodes containing sentences with the word Azithromycin in them highlighted green.

sentences together. A simple K-nearest neighbors approach with a cosine similarity metric was used as a comparison. In order to use a nearest neighbors approach, a starting sentence is needed. This sentence was chosen by selecting a random sentence from the list of sentences containing both hydroxychloroquine and coronavirus. The sentence used for these queries is: 'Antimalarial prophylactic drugs, such as hydroxychloroquine, are believed to act on the entry and post-entry stages of SARS-CoV (severe acute respiratory syndrome-associated coronavirus) and SARS-CoV-2 (severe acute respiratory syndrome coronavirus 2) infection, likely via effects on endosomal pH and the resulting under-glycosylation of angiotensin converting enzyme 2 receptors that are required for viral entry.' This sentence contains references to both the actual virus, as well as an example drug and how it may interact with the virus itself, allowing for a variety of sentences to be found to be similar.

The five closest sentences returned by the nearest neighbors based approach are found below in Table 4.2. In general, these sentences focus on how the coronavirus uses the ACE-2 receptor to enter into host cells.

A sample of the sentences found in the same leaf node of the THD as the input sen-

tence are found below in Table 4.3. The sentences in this leaf node seem to have more focus on the drug treatment portion of the input sentence and focus less on the mechanism by which the drugs may be working.

This example shows that the groupings performed by the THD group sentences in a different manner than using a simple clustering method and as a result may add more value for finding relevant sentences.

In order to further explore the effectiveness of the groupings generated by a THD, a set of very similar sentences were seeded in the construction of the THD. These sentences were then queried to determine how many ended up in the same leaf node, and their overall distance. These sentences were then compared to a set of vastly different sentences to ensure they were in distinct sets of nodes within the THD. The results of this test are discussed below. One set of input sentences was a group of sentences containing copyright information. This set of 5 sentences had an average cosine similarity of 0.0353. These sentences were found to all fall into a single leaf node within the THD structure. The set of sentences was then expanded to 10 sentences. Upon expanding to 10 sentences, the average cosine similarity between the sentences was 0.1137. These sentences are spread out over more leaf nodes, but all share a common parent and are segmented out extremely quickly. A majority of these sentences still occur in one leaf node with the rest in individual leafs. These sentences were compared to another set of 5 short sentences regarding hydroxychloroquine. The 5 closest sentences were found to have an average cosine similarity of 0.0501. These sentences were again found to exist in a single leaf node. The number of sentences was increased to 10, and the average similarity of these sentences increased to .0866. Of the 10 sentences, 9 of them fell into one leaf node, and the remaining sentence fell into another leaf node. The average similarity between the two sets of sentences was 0.4959, showing they are vastly different. In the THD structure, the copyright sentences appear in a leaf node that branches very early in the segmentation process, whereas the sentences regarding hydroxychloroquine occur in a leaf node at the end of the segmentation process. This

shows the THD is able to capture both the similarity between sentences by grouping them in the same node, and dissimilarity with nodes in entirely different branches.

The THD constructed using a correlation metric consisted of a single main branch with 5 small single leaf branches. This indicates poor performance and an inability to segment the data. Despite these few branches, this THD had many more groups than the previous two THDs discussed. This further reinforces that the THD was unable to properly segment the sentence embeddings when using a correlation metric. Figure 4.16 shows the structure of this THD.

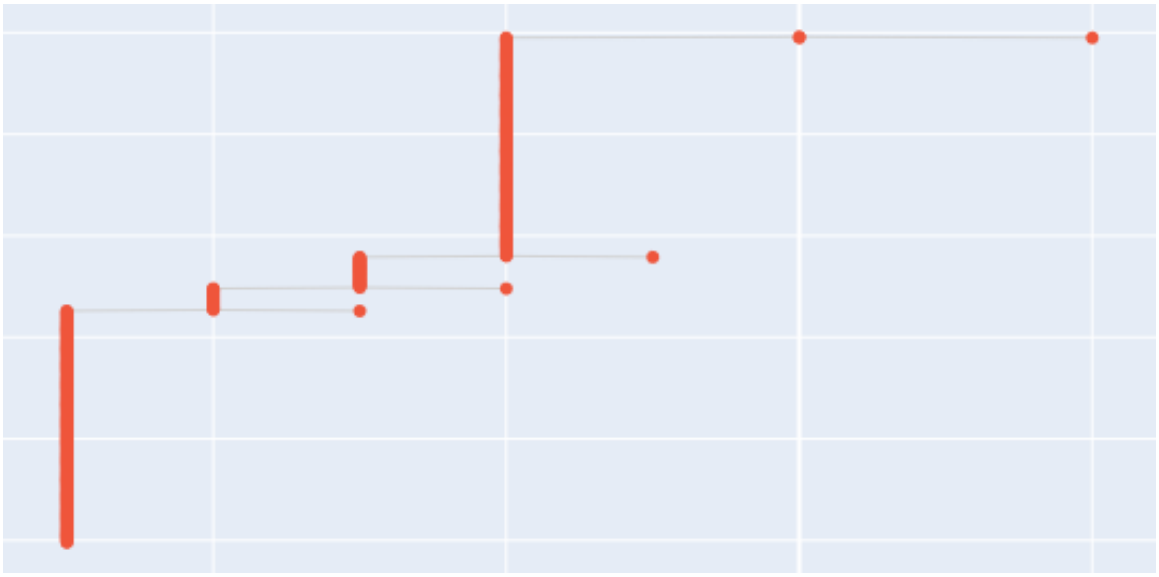

Figure 4.16: THD constructed from 300-dimensional sentence using a correlation metric.

Similarly to the THD constructed using a correlation metric, the THD constructed using cosine with a resolution correction step had a greatly increased number of resulting groups. This larger number is expected, and indicates that the resolution correction parameter may have allowed for branches to segment more effectively than in the architecture without this parameter. The resulting structure for this THD is found below in Figure 4.17. This figure shows that the increased number of groups are in fact a result of an extremely long main branch, with relatively few branches off of it. This structure mimics that of the THD constructed using a correlation metric, and indicates that the resolution correction

step prevented effective segmentation from occurring. The resulting number of groups and

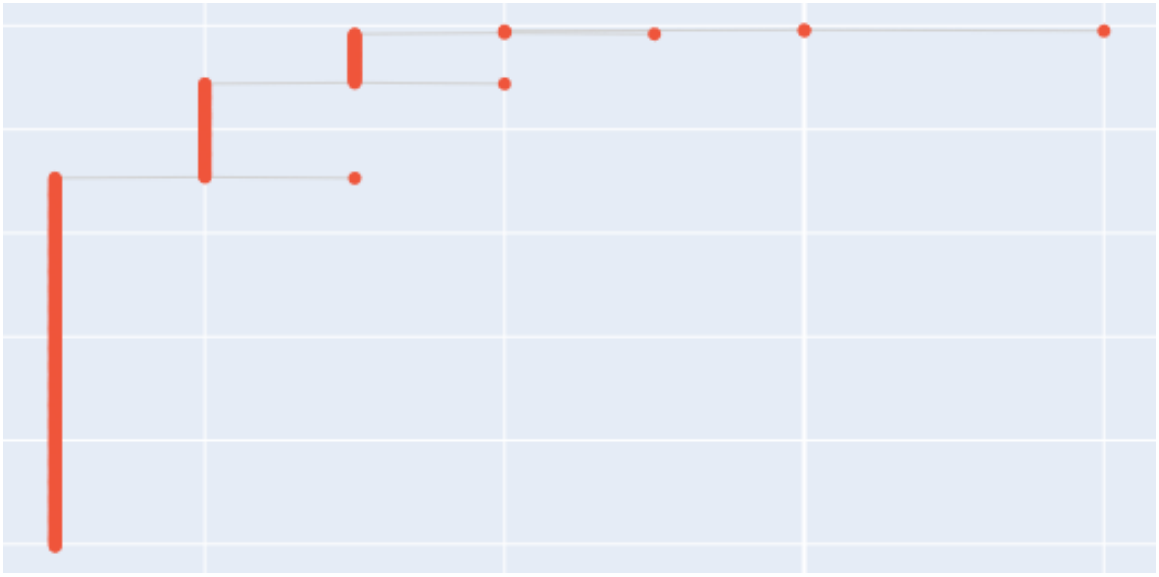

Figure 4.17: THD constructed from 300-dimensional sentence using a cosine metric with a resolution correction step.

average group sizes for the resulting THDs that were described above can be found below in Table 4.4. The number of groups in a THD is defined as all groups in the THD that meet the group threshold requirement set when constructing a THD. In this table the number of leaf nodes is calculated by counting the number of groups in the THD that have no further segmentation and meet the group threshold of 100 set when initially constructing the THDs. These leaf nodes are a subset of the entire set of groups in the THD. Leaf size is defined as the number of distinct sentences in a given leaf node. As such, maximum leaf size is the number of sentences in the largest leaf node. As shown in this table, the THDs constructed using a cosine similarity metric and neighborhood lenses without resolution correction had similar numbers of nodes and group sizes. This is expected due to the similarity between the two architectures. Also shown in the table is that the THD constructed using resolution correction at branch points resulted in a much larger number of groups. This is expected, and indicates that the resolution correction parameter was effective in enabling further segmentation of branches in the THD. This increase in group number is also apparent in the

THD constructed using the correlation distance metric instead of a cosine metric. This may be a result of improved segmentation, but could be due to other reasons.

As the document embeddings obtained from the THDs are based on the membership of sentences in the groups of a THD, Table 4.4 also shows the dimensionality of the document embeddings obtained from each individual THD. This shows that although the resolution correction step improves the quality of branches in a THD, it also potentially lowers the THD's usefulness for other tasks such as document embeddings. These embeddings are still used as input to a hierarchical clustering technique in comparison to the embeddings from THDs containing smaller numbers of groups.

The resulting statistics for the THD constructed using a correlation metric indicates that the resulting THD suffered from poor segmentation, and instead shed a large amount of singletons very quickly that were not tracked, and were instead treated as outliers that did not have meaningful data. This resulted in a very small number of leaf nodes, in which all but one had an extremely small number of sentences. These leaf node sizes indicate that the THD was unable to properly segment the data, and reached a maximum resolution value before any meaningful segmentation had occurred on the main branch of the THD. This indicates that the correlation metric is an ineffective method when applied to the sentence embeddings used in this paper.

In comparison, the two THDs constructed using a cosine metric without any form of resolution correction resulted in a more appropriate branching structure with a relatively large number of moderately sized leaf nodes. This indicates that the THD was able to segment the data much more effectively than in the previously discussed THD. These THDs do shed a vast majority of the sentences in the dataset as singleton outliers as the THD progresses, though this occurs more gradually, rather than in a single occurrence as in the THD constructed using correlation as a metric. This indicates that a lower group threshold may be beneficial for THDs constructed on sentence embeddings in order to track much more specific sentences that may not have sufficient neighbors near them. Though a large

number of singletons were shed in these THDs, they were still effective at returning meaningful sentences from a variety of queries. This indicates the most important sentences in documents may be retained over less important sentences, allowing for a compressed view of documents.

As the THD constructed using cosine similarity as its metric on the 300-dimensional sentence embeddings seemed to have the best overall structure and groupings, it was selected for further querying based on metadata and more similarity searches. A few samples of these searches are discussed below.

An example search based off of only journal title metadata is shown below in Figure 4.18. This shows that the content of the journal Virus Research primarily falls into specific groups of the THD, which likely indicate those branches of the THD are related to topics commonly covered by Virus Research.

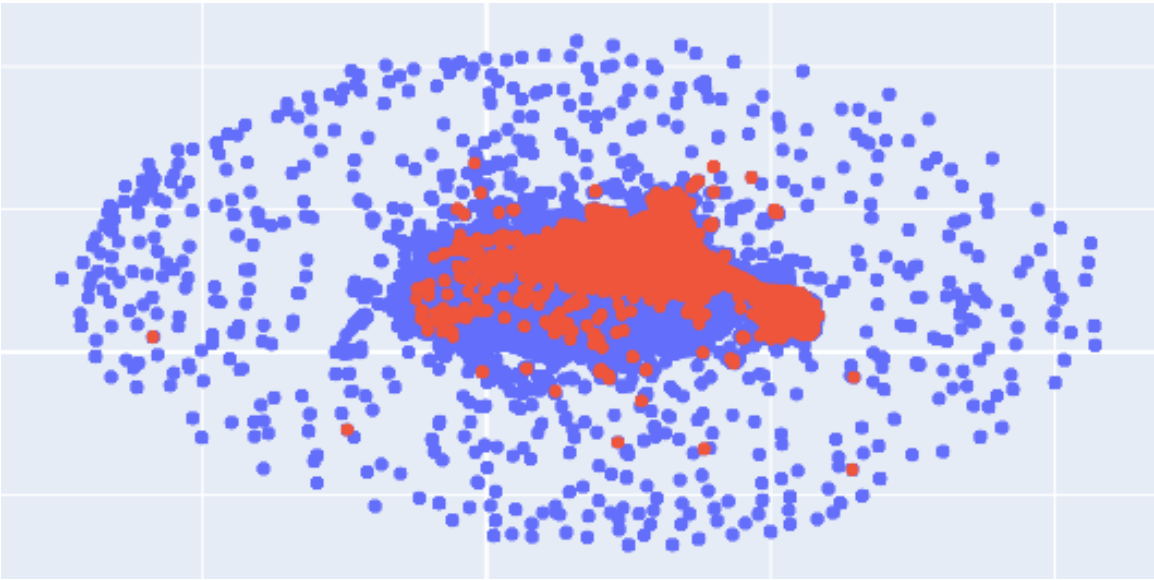

Figure 4.18: Sentence plot for all sentence embedded in 300 dimensions. Sentences from papers published in the journal Virus Research are highlighted orange.

Figure 4.19 shows another metadata based search that includes year and journal information. This search shows the groups of the THD that contain sentences from the Virus Research between 2019 and 2020. These resulting groups and locations are similar to the

overall groups found in the query based solely on the name of the journal. This indicates that the topics covered by papers in the journal have not experienced any major shift over time. These two searches show the usefulness for using THDs to assess topic trends over time, even though time data was not included in the initial input to the THDs.

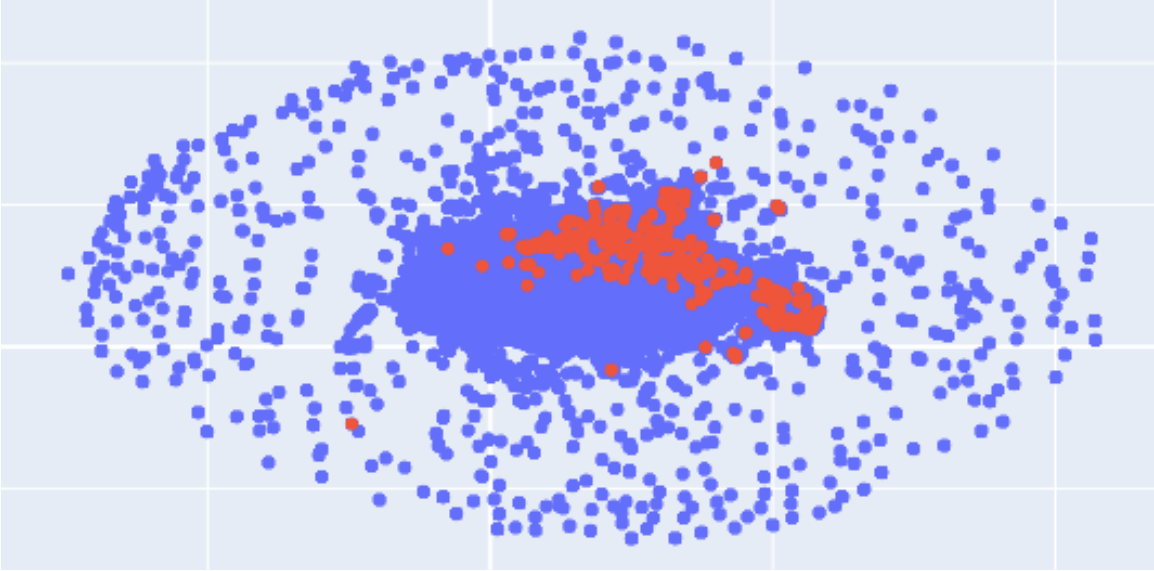

Figure 4.19: Sentence plot for all sentence embedded in 300 dimensions. Sentences from papers published in the journal Virus Research since 2019 are highlighted orange.

The results of a search for sentences most similar to the words coronavirus and angiotensin are found below in Figure 4.20. These sentences group into three small clusters in the original embeddings, and end in a similar number of distinct groups of the THD as well.

The results of a search for sentences containing the words coronavirus and angiotensin are found below in Figure 4.21. The resulting groups that are highlighted in the THD are very similar to the groups highlighted by the similarity based search. This indicates that the resulting embeddings accurately capture the content of the sentences, and can be used effectively for queries in either form.

To further show the usefulness of queries based on similarity, two potential use cases

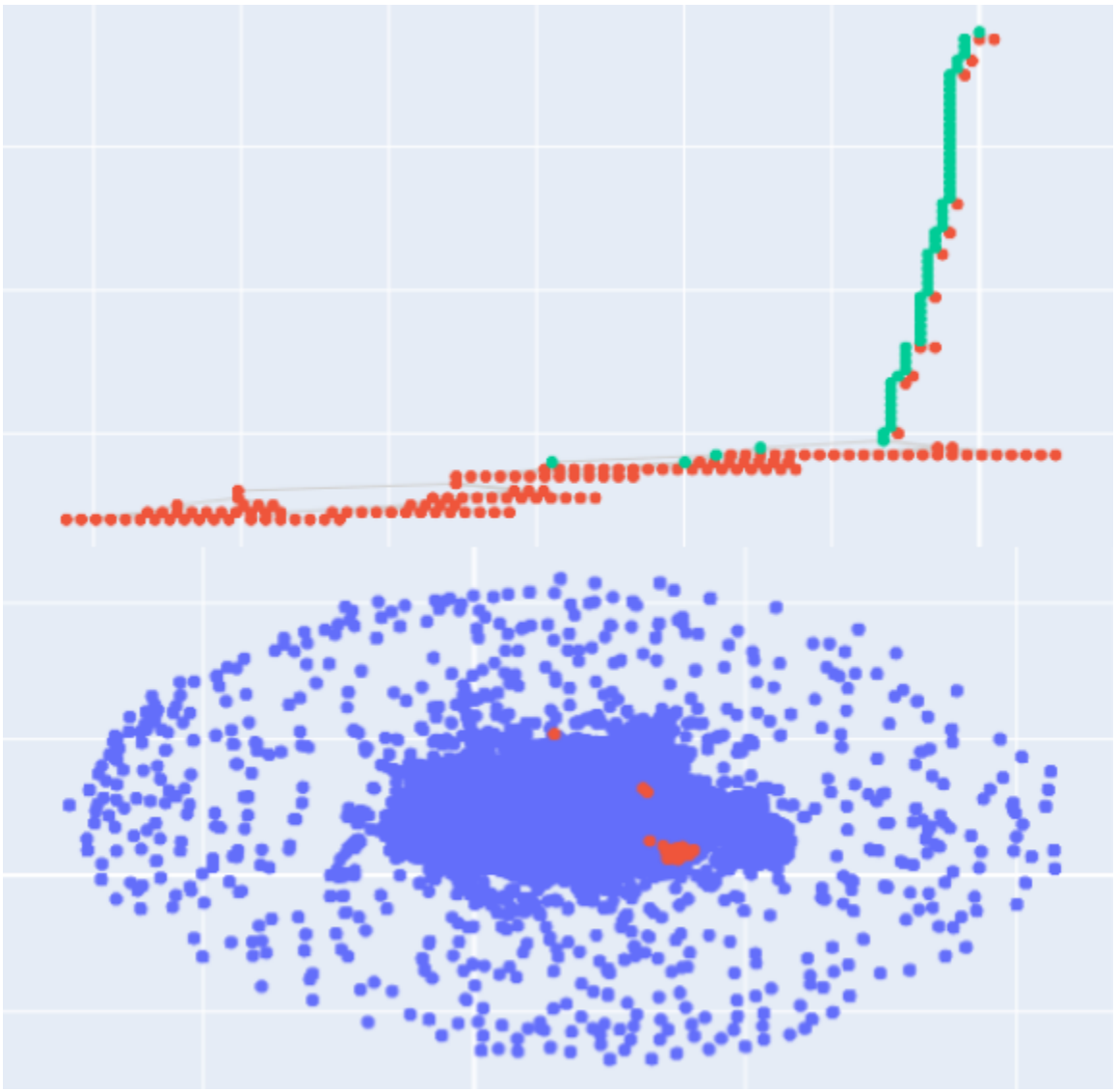

Figure 4.20: The upper portion of this figure shows the THD constructed on 300-dimensional embeddings with a cosine metric and neighborhood lenses. Nodes highlihted green contain sentences closest to a sentence with just the words angiotensin and coronavirus. The bottom figure shows all sentences with 300-dimensional embeddings. Nodes highlighted orange in this portion are sentences that are closest to a sentence containing only the words angiotensin and coronavirus.

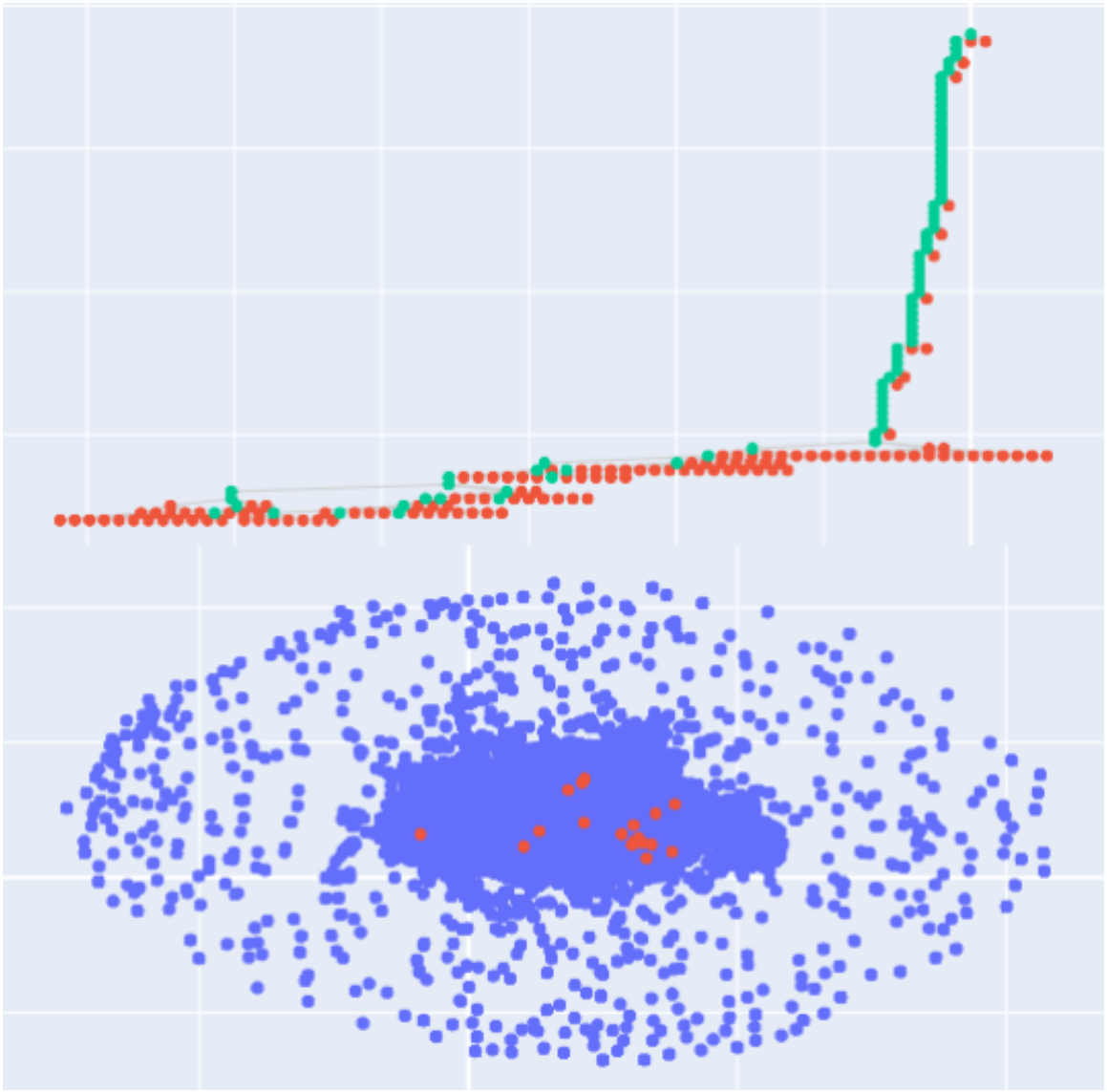

Figure 4.21: The upper portion of this figure shows the THD constructed on 300-dimensional embeddings with a cosine metric and neighborhood lenses. Nodes highlihted green contain sentences with the words angiotensin and coronavirus. The bottom figure shows all sentences with 300-dimensional embeddings. Nodes highlighted orange in this portion are sentences that contain the words angiotensin and coronavirus.

using these queries are explored below.

One potential use case for the dashboard designed above is to find relevant documents for a literature survey, help validate a hypothesis or to find potential alternative techniques. This can be done by performing a query based on an input hypothesis to find the most similar sentences and the leaf nodes they exist in. Users can then assess the value of each sentence and note the corresponding document ID for these sentences. These document IDs can then be used to look up the document the sentences are found in for users to read. An example of this use case is explored below to find documents that may discuss alternative drug treatments for the 2019 coronavirus epidemic.

To begin this use case, the 100 sentences most similar to a sentence containing just the words coronavirus and hydroxychloroquine were found and highlighted within the THD structure. This result is found below in Figure 4.22.

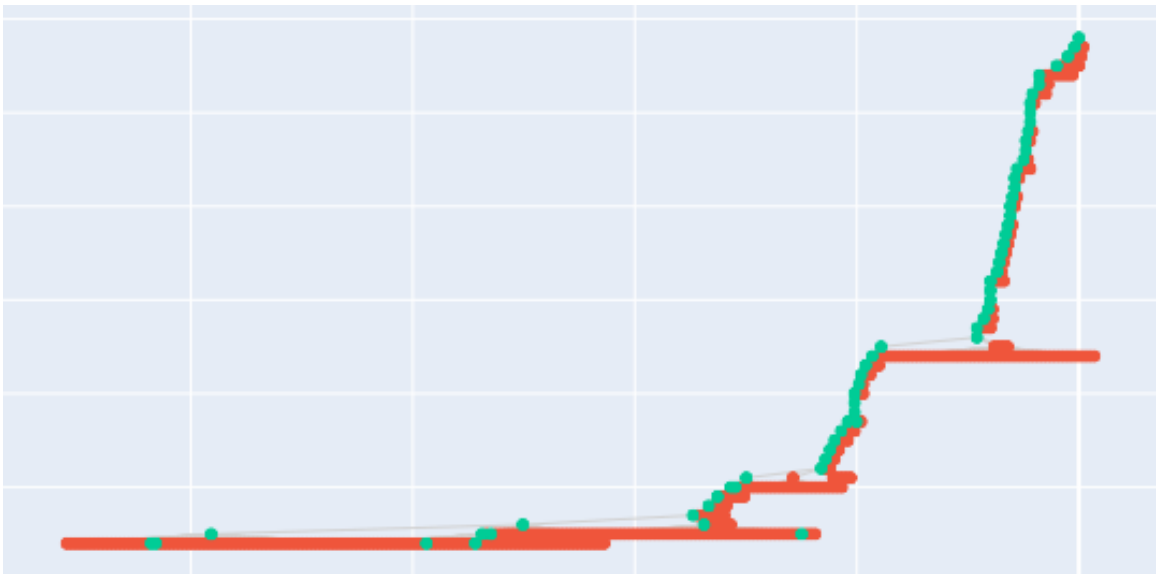

Figure 4.22: THD highlighted by nodes containing one of the 100 sentences most similar to a sentence containing only the words ‘hydroxychloroquine’ and ‘coronavirus’.

Next, the sentences within the far left node highlighted by the search were analyzed to find sentences that may be indicative of a relevant paper. All sentences containing either ‘chloroquine’ or ‘hydroxychloroquine’ were analysed and used to find potential documents.

These sentences existed in 7 unique documents within this node. Of those documents, 6 were directly related to treatment of coronaviruses and 1 was related to the usage of chloroquine on influenza viruses. Of the documents related to treating coronaviruses, 3 were a survey of current and potential treatments including alternatives to hydroxychloroquine such as Zinc and Remdesivir. The remainder of the documents specifically discuss the usage of hydroxychloroquine as a treatment.

Another use case exists for finding potential collaborators for research based on the content of their published papers. This use case functions similarly to the use case discussed above for finding relevant documents and requires the user to input a query to the dashboard. When performing the query, an option is given to return metadata about the sentences returned by the query. When this option is checked, the dashboard dynamically populates plots based on the top 10 documents based on number of sentences returned. One of these plots is a pie chart distribution of the authors of the documents containing the sentences. This plot can be used by itself to find potential collaborators. Alternatively these authors can be used to perform further metadata based queries to quickly assess how diverse the research done by any author is. An example exploring this use case to find potential collaborators for researching the effectiveness of remdesivir as a potential treatment for coronavirus is shown below. Figure 4.23 shows the THD structure highlighted by nodes that contain any of the sentences most similar to a sentence just containing the words remdesivir and coronavirus.

Figure 4.24 shows the corresponding year histogram for these sentences showing that these sentences are predominantly within the last year with research that may be relevant spanning back until 2005.

The author distribution for the 10 most similar sentences was analysed and was found not to have any prolific authors, with all included authors having an equal number of documents published, which is expected as the usage of remdesivir for treating coronavirus is a

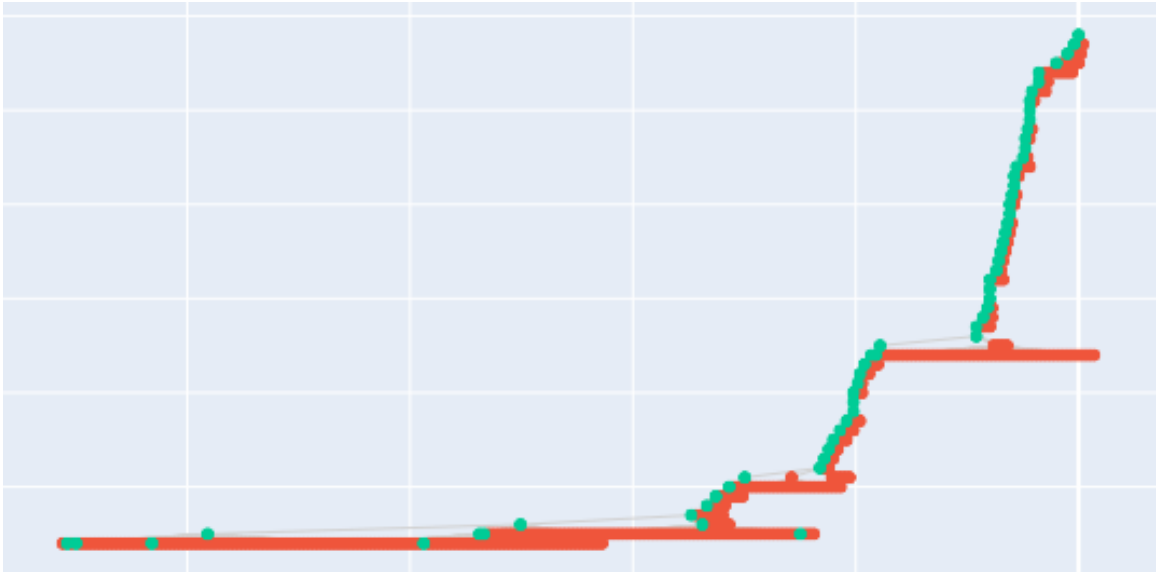

Figure 4.23: THD highlighted by the 100 sentences most similar to a sentence containing only the words ‘remdesivir’ and ‘coronavirus’.

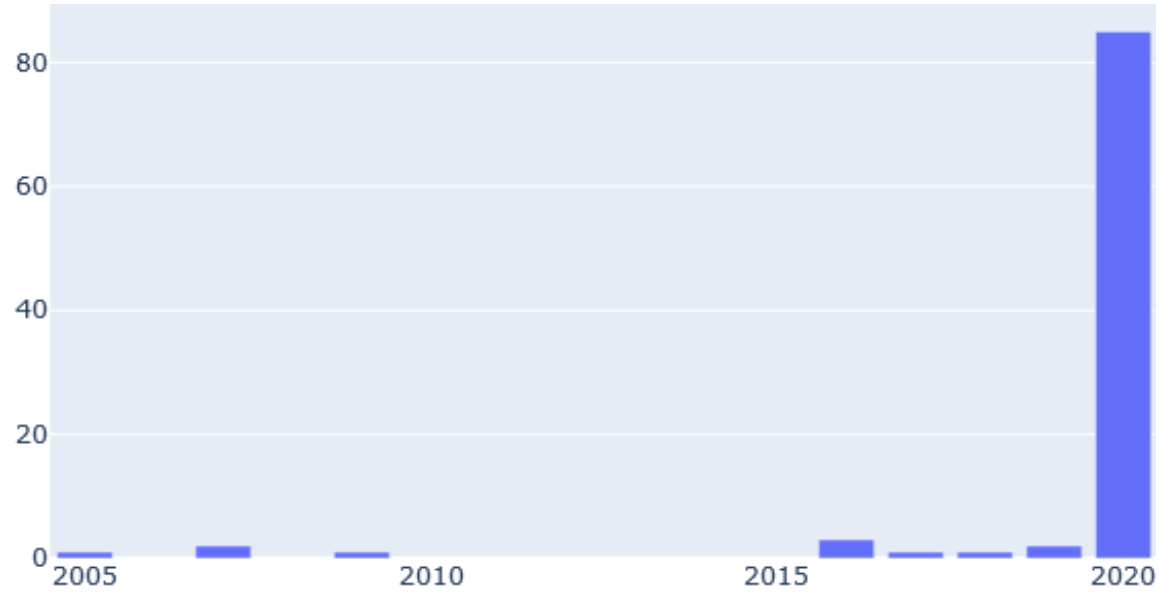

Figure 4.24: Year histogram showing the number of papers published in each year that contain one of the 100 sentences most similar to a sentence containing only the words ‘remdesivir’ and ‘coronavirus’.

relatively new concept. As there does not appear to be any prolific authors in this area, one was chosen at random to use for this example. This author was Laura Bauer.

Figure 4.25 below shows the THD structure highlighted by nodes that contain sentences found in documents authored by Laura Bauer. These nodes are found in a relatively similar location to the nodes highlighted by the first query in Figure 4.23.

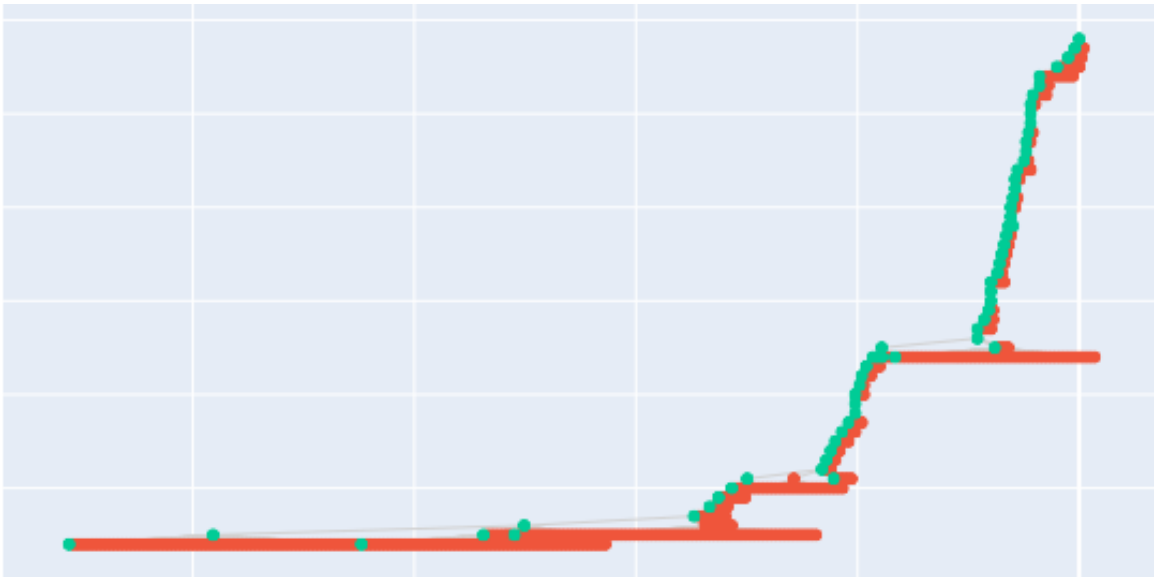

Figure 4.25: THD highlighted by nodes containing sentences from papers published by the author Laura Bauer.

This indicates that this author may be only publishing recently, which can be confirmed by looking at the year histogram for the author found below in Figure 4.26.

The resulting silhouette scores for the clusters obtained using the document embeddings generated from the THDs can be found below in Table 4.5. As shown in the table, the embeddings obtained from THDs constructed on higher-dimensional sentence embeddings obtain much better scores.

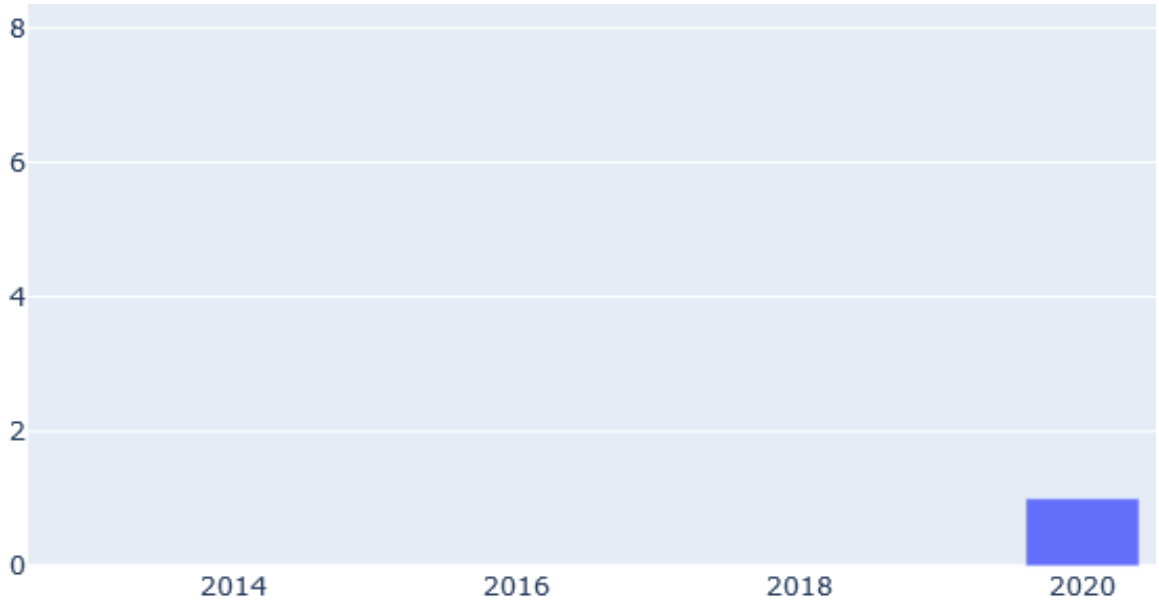

Figure 4.26: Year histogram showing the number of papers published per year by the author Laura Bauer.

Sentence	Distance
'A recent study on SARS-CoV-2 infection showed that all the 41 patients had pneumonia and manifestations of a critical respiratory ailment similar to severe acute respiratory syndrome (SARS) coronavirus and was associated with ICU admission and high mortality .It is known that coronaviruses such as human SARS-CoV and, bat SARS-like CoV SL-CoVZXC21 utilizes Angiotensin-converting enzyme 2 (ACE2), as their receptor and recent reports suggest that SARS-CoV-2 also uses the identical receptor for entry into the host cell .'	0.8006
'A neuroradiologist should be aware of the potential mechanisms involved in the neuropathogenesis of this virus, as we begin to see cases with abnormal brain scans emerging from all parts of the world.The causative agent severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) utilizes the Angiotensin Converting Enzyme 2 receptor (ACE2) for entry into host cells, and causes a severe clinical syndrome manifested primarily as a respiratory tract infection [3] .'	0.8078
'In a similar manner to the coronavirus that caused Severe Acute Respiratory Syndrome (SARS-CoV), the novel coronavirus responsible for COVID-19 utilizes ACE2 on the surfaces of epithelial cells to bind and gain entry to infected cells [4, 5] .Diabetes and conditions such as hypertension are associated with activation of the renin-angiotensin system in different tissues.'	0.8188
'Under these exigent conditions, the administration of the already developed safe drugs should be the smartest shortcut.TMPRSS2 is a serine protease that primes the spike protein of highly pathogenic human coronaviruses, such as severe acute respiratory syndrome-related coronavirus (SARS-CoV) and Middle East respiratory syndromerelated coronavirus (MERS-CoV), and facilitates its entry into the host cell.'	0.8302
'Unexpectedly, it was very recently identified as a functional receptor for the coronavirus (CoV) that causes the severe acute respiratory syndrome (SARS), serving as the cellular entry point for the SARS virus .'	0.8402

Table 4.2: Sentences with embeddings closest to the embedding for the sentence 'Antimalarial prophylactic drugs, such as hydroxychloroquine, are believed to act on the entry and post-entry stages of SARS-CoV (severe acute respiratory syndrome-associated coronavirus) and SARS-CoV-2 (severe acute respiratory syndrome coronavirus 2) infection, likely via effects on endosomal pH and the resulting under-glycosylation of angiotensin converting enzyme 2 receptors that are required for viral entry.' with corresponding distances between the embeddings.

Sentence
'Treatment must be accompanied by antiviral treatment (lopinavir/ritonavir or remdesivir + chloroquine/hydroxychloroquine) and/or steroid (dexamethasone) [30] .In our low/medium intensity ward, we have drawn up a protocol of patient's examinations to be requested both at the time of admission and during the course of the hospital stay, based from evidence and clinical judgment.We present in details in Table 3.'
'[7] Treatments used to date have been proposed to work by: 1) limiting entry into ciliated bronchial epithelial cells (N-acetylcysteine, heparin, meplazumab, umifenovir, hydroxychloroquine), 2) inhibiting viral replication (interferon-α/β, ritonavir/lopinavir, oseltamivir, ganciclovir, ribavirin, favipiravir, remdesivir, danoprevir), 3) preventing viral dissemination via antibody-mediated neutralization by increasing SARS-CoV-2-specific antibodies (convalescent plasma) or non-specific antibodies (IVIg, thymopentin), 4) strengthening a weakened immune response with immunostimulants (interferon-α/β, thymosin-α-1), 5) preventing a hyper-immune response with immunosuppressants (corticosteroids, hydroxychloroquine, IVIg), or 6) controlling a hyper-immune response (corticosteroids, tocilizumab).'
'Concerns about the generalizability of these findings include: 1) lopinavirritonavir was started late in the disease course (median of 13 days after symptom onset), while animal and human data from other coronaviruses suggested that early antiviral initiation is important; 2) higher baseline viral loads in the treatment arm, though there was no difference in the change from baseline across the two arms; and 3) a high mortality rate in this cohort, perhaps limiting ability to extrapolate these data to other, less sick patients. There are additionally several published case reports and case series describing patients treated with lopinavir-ritonavir, including a series of 36 children in which 14 were treated with lopinavir-ritonavir.All patients recovered, and no comparisons were made between treated and untreated patients (110) .'
'Based on the documented inhibition of coronavirus replication, Remdesivir and 136 K22 were included as a positive control [8, 12] .'
'Chloroquine and hydroxychloroquine have now been permanently included, alongside antiviral drugs, in protocols for the treatment of COVID-19 pneumonia [10] .'

Table 4.3: Sample of sentences within a THD node containing the sentence 'Antimalarial prophylactic drugs, such as hydroxychloroquine, are believed to act on the entry and post-entry stages of SARS-CoV (severe acute respiratory syndrome-associated coronavirus) and SARS-CoV-2 (severe acute respiratory syndrome coronavirus 2) infection, likely via effects on endosomal pH and the resulting under-glycosylation of angiotensin converting enzyme 2 receptors that are required for viral entry.'

THD Parameters	Number of Groups	Number of Leaf Nodes	Minimum Leaf Size	Maximum Leaf Size	Average Leaf Size
100 Dimensions Cosine	250	114	100	11826	634.92
300 Dimensions Cosine	206	130	101	5786	446.35
300 Dimensions Correlation	504	6	109	689828	115122.5
300 Dimensions UMAP	431	371	100	33771	806.6
300 Dimensions Corrected	503	6	108	682982	113982.8

Table 4.4: THD structure statistics

THD Parameters	Silhouette Score (6 Clusters)
TF-IDF (Baseline)	0.0933
100 Dimensions Cosine	0.2692
300 Dimensions Cosine	0.5645
300 Dimensions Correlation	0.5659
300 Dimensions UMAP	0.3247
300 Dimensions Cosine Corrected	0.5626

Table 4.5: Table of silhouette scores for document clusters obtained from various THD settings.

Conclusions

The resulting sentence embeddings appear to be effective when used for tasks that require similar sentences. Based on this evaluation, and the plots of the resulting embeddings, the method used to generate sentence embeddings in this paper appears to be an effective alternative to more complex methods such as BERT embeddings. This indicates that highly complex models are not always necessary for tasks that are primarily focused on the content of sentences.

Based on the resulting THDs, it can be concluded that they are an effective method for visualizing and assessing the quality of set of sentence embeddings. As is evident from the plots of the THD structures, a cosine metric provides the best quality segmentations that allow for easy understanding of individual groups. The branching of these THDs also indicates that the sentence embeddings obtained from averaging word embeddings are capable of capturing sufficient semantic meaning to influence branching when used in THDs. The resulting THDs also indicate that although resolution correction is usually useful in large datasets, it is detrimental when applied to sentence datasets. This may be due to the fact that the dataset is related to one overarching topic, resulting in a rather homogenous structure that does not segment until it is viewed at extremely high resolutions in THDs. It is likely that further work with more varied datasets using a similar approach may result in better segmentation with resolution correction than was observed for this dataset.

The THDs constructed in this thesis were able to group semantically similar sentences

in a hierarchy as shown specifically by discovery of a drug treatment branch in the THD. In addition the leaf nodes of THDs provide an effective means of finding similar sentences that works as well as K-means clustering.

In general, THDs are sensitive to metric and lens parameter choices. These parameters can drastically affect the quality of the structure of the resulting THDs, and as a result their usefulness. Some metrics and lenses produce good structure for most datasets, and as such are a good starting point. These include cosine and euclidean metrics as well as neighborhood lenses. The cosine metric was found to produce the best resulting THDs for the dataset used, and neighborhood lenses were found to be an effective metric. In addition, UMAP lenses were also found to produce well structured THDs.

Further experimentation with a correlation metric did not produce well structured THDs, as such, this metric should be avoided when working with sentence embedding datasets. Similar datasets based on sentence embeddings will likely perform well when used to construct THDs using a cosine metric and neighborhood or UMAP lenses, regardless of the size of the corpus, as the number of data points in a dataset has a minimal impact on the performance of a THD.

The dashboard built in this paper allowed for quick and efficient querying of the entire dataset of over 100,000 documents. This efficient dashboard system allowed for quick assessment of the quality of THDs and further analysis of the underlying dataset to uncover trends over time with respect to various topics and journals. Further, this dashboard would allow users to uncover potentially valuable information to quickly improve research and form hypotheses for further research, reducing the time spent reading unnecessary documents that may be only peripherally related to a topic.

The document embeddings generated from the resulting THDs appear to be an alternative method to using TF-IDF vector representations for document clustering and document retrieval tasks, which indicates that there is room for improvement in these tasks. Further these results using the naive K-Means approach indicate that further improvement may oc-

cur when these embeddings are used in tandem with more complex clustering techniques. The document embeddings obtained from the THDs constructed on higher-dimensional sentence embeddings obtain much higher scores, indicating that the additional dimensions allow the THDs to more accurately segment out topics, which can be used to improve the separation of the resulting documents into more clearly defined clusters. The relatively high scores for the poorly segmented THDs may be a result of the similar number of leaf nodes and clusters chosen, resulting in clustering based on which leaf node has a higher value. This is likely not the case for the THD constructed using cosine with no model correction, due to the relatively large number of leaf nodes in comparison to the number of document clusters.

Future Work

In the future, more work could be done to add additional functionality to the dashboard initially designed in this paper. The ability to further refine queries using combinations of all methods may allow users to perform more useful queries quicker, by allowing for searches in specific time ranges or journals. In addition further THD parameters could be explored to determine if there are more effective combinations of lenses and metrics that improve the resulting THD structures and their effectiveness for more complex tasks such as document clustering. THDs could also be constructed on datasets of sentence embeddings generated using different methods, such as weighted averaging, or even more complex models such as BERT to determine if these more complex models would provide better resulting THD structures. Further, a series of THDs could be constructed based on a variety of different sentence embedding techniques and combined to potentially capture more information than any singular sentence embedding technique is capable of capturing in a vacuum. Additionally, improvements to the method used in downsampling the sentences used for THDs may result in higher quality THDs. These could potentially be further improved by constructing THDs on the entire dataset of over 7 million sentences. These THDs may uncover additional topic areas in the dataset that were excluded due to issues with the downsampling technique chosen.

A further potential application of THDs would be to use them to generate sentence embeddings themselves. This could be done by generating word embeddings by any method and then constructing a THD based on these word embeddings. A sentence could be rep-

resented by the nodes in the corresponding THD that the words of the sentence occur in. This could allow for interesting groupings of sentences containing completely disjoint sets of words, but whose words end up in the same leaf nodes due to their similarity.

Bibliography

[1] Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*, 2016.

[2] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *2008 eighth IEEE international conference on data mining*, pages 3–12. IEEE, 2008.

[3] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[4] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[7] Adrien Bougouin, Florian Boudin, and Béatrice Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. 2013.

[8] Kyle Brown, Derek Doran, Ryan Kramer, and Brad Reynolds. Heloc applicant risk performance evaluation by topological hierarchical decomposition. *arXiv preprint arXiv:1811.10658*, 2018.

[9] Qingyu Chen, Yifan Peng, and Zhiyong Lu. Biosentvec: creating sentence embeddings for biomedical texts. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–5. IEEE, 2019.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[11] Tamal K Dey, Facundo Mémoli, and Yusu Wang. Multiscale mapper: topological summarization via codomain covers. In *Proceedings of the twenty-seventh annual acm-siam symposium on discrete algorithms*, pages 997–1013. SIAM, 2016.

[12] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

[13] Richard T Freeman and Hujun Yin. Adaptive topological tree structure for document organisation and visualisation. *Neural networks*, 17(8-9):1255–1271, 2004.

[14] Hui Guan, Wen Tang, Hamid Krim, James Keiser, Andrew Rindos, and Radmila Sazdanovic. A topological collapse for document summarization. In *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2016.

[15] Thomas Hofmann. Probabilistic latent semantic analysis. *arXiv preprint arXiv:1301.6705*, 2013.

[16] James Jardine and Simone Teufel. Topical pagerank: A model of scientific expertise for bibliographic search. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 501–510, 2014.

[17] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[18] Michael Steinbach George Karypis, Vipin Kumar, and Michael Steinbach. A comparison of document clustering techniques. In *TextMining Workshop at KDD2000 (May 2000)*, 2000.

[19] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational linguistics*, 32(4):485–525, 2006.

[20] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[21] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[22] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE, 2015.

[23] Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. Topic-link lda: joint models of topic and author community. In *proceedings of the 26th annual international conference on machine learning*, pages 665–672, 2009.

[24] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 257–266, 2009.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[26] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[29] Thuy Dung Nguyen and Min-Yen Kan. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer, 2007.

[30] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[31] Sun Park, Dong Un An, and Choi Im Cheon. Document clustering method using weighted semantic features and cluster similarity. In *2010 Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, pages 185–187. IEEE, 2010.

[32] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[33] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA, 2003.

[34] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[35] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[36] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. *SPBG*, 91:100, 2007.

[37] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

[38] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[39] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, et al. Cord-19: The covid-19 open research dataset. *ArXiv*, 2020.

[40] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.

[41] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI global, 2005.

[42] Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. Legal judgment prediction via topological learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3540–3549, 2018.